RoboCup Rescue 2016 Team Description Paper Team GETbots (Germany)

Dirk Fischer, Bärbel Mertsching, Hossein Mirabdollah, Mahmoud A. Mohamed, Muhannad Mujahed, Daniel Nickchen, and Mawe Sprenger

Info

Team Name:	GETbots
Team Institution:	Paderborn University
Team Leader:	Dirk Fischer
Team URL:	http://getwww.uni-paderborn.de/getbots

RoboCup Rescue 2016 TDP collection:
https://to-be-announced.org

***Abstract*—In this paper, we describe our approach for the participation in the 2016 RoboCup Rescue League competition. Two different robots will be deployed: (i) A P3-AT based platform for autonomous operation, and (ii) a teleoperated platform based on the Jaguar V4 for tasks requiring high mobility. Our software architecture is based on the open source framework ROS [1][2] and comprises various components such as SLAM, exploration, visual victim detection and localization, etc. Many components have been developed by students of GET Lab. We have been testing and prototyping the components mainly in our simulation environment SIMORE [3].**

***Index Terms*—RoboCup Rescue, Team Description Paper, Robot Control, Autonomous Exploration, 3D Mapping.**

I. Introduction

GET Lab is an interdisciplinary research group at the University of Paderborn with a research focus on cognitive systems. The team GETbots was established in GET Lab in 2007. It comprises master and doctoral students from a variety of disciplines. Its research objectives are related to autonomous exploration of unknown environments, creation of environment maps, search for victims and assessing their health situations, simulation environments, user interface design, and robot control architectures.

Since 2008 the team GETbots has successfully participated in the RoboCup German Open competitions and achieved numerous honors: second place in the overall competition in 2009, third place in 2008, 2011 and 2012, “Best in Class Mobility” award in 2013 and 2014, and “Best in Class Manipulation” award in 2015. This year we have been enhancing software components already used in previous years. In order to deal with new challenges which might appear in future events, we will incorporate recently developed features as there are 3D mapping, 3D object detection and classification, multi robot exploration, as well as intuitive control for teleoperated robots.

Similar to the last competition, we plan to use two different robots. The first one, called GETbot, is based on a P3-AT platform from Mobilerobots [4] (see Fig. 1). The hardware configuration of the robot is presented in table I in appendix

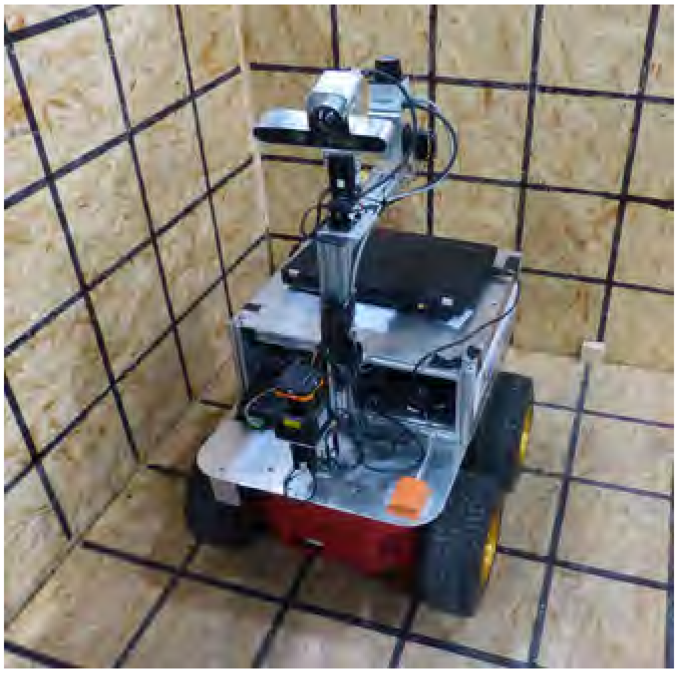


Fig. 1: Autonomous robot *GETbot*.

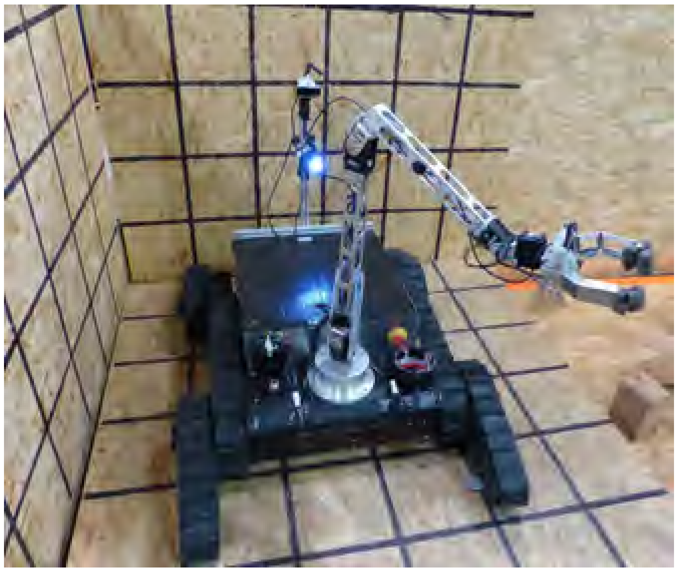


Fig. 2: Teleoperated robot *GETjag*.

B-A. Furthermore, we will deploy a robot with four articulated arms called GETjag (see Fig. 2). The robot is highly mobile, able to climb stairs, and perform grasping and inspection tasks with a self-designed manipulator arm platform. The robot's configuration is listed in table II in appendix B-A.

A. Improvements to Previous Contributions

Our victim detection framework has used a combination of thermal, visual and depth cues to detect victims. Currently, we are working on increasing the reliability of our framework and reducing false positives by applying an advanced object detection algorithm. The new algorithm extracts human body features (e.g. temperature, face, body, etc.) from different sensors and uses the AdaBoost machine learning technique [5] to detect victims.

During the last competitions, we basically had made use of RGB-D data to analyze the structure of the underground so that the traversable paths can be detected. Currently, we are integrating new algorithms based on an RGB-D sensor for 3D object recognition, advanced obstacle detection and 3D reconstruction. Taking 3D structures of the environment into account will assist our robots to perform rescue operations more efficiently.

In our previous software architecture, a hierarchical state machine was used to control the behavior of our autonomous robot. Nevertheless, using the state machine can give rise to different difficulties. For instance, adding a new behavior or additional constraints often involves modifying several states and transitions. Hence, we currently work on using the idea of goal oriented action planning [6] for the autonomous control and decision making. We are confident that this will make our autonomous system more flexible and easier to adapt to new challenges.

II. System Description

A. Hardware

1) RGB-D Sensors: Both of our robots are equipped with an ASUS Xtion Pro Live depth camera producing RGB-D data with a frame rate of up to 30 Hz. Recently, we have purchased a Kinect V2 depth sensor and plan to replace one of the Xtion cameras with it. As the Kinect V2 sensor conducts depth measurements based on the time-of-flight principle and has an RGB resolution of 1080p, it apparently produces more accurate RGB-D data than the Xtion sensor working based on the structured-light principle. Additionally, the Kinect V2 sensor has a larger field of view both horizontally and vertically as well as a larger detection range than the Xtion. Thus, we expect to achieve enhancements in our 3D analysis module (see II-B4 for details).

2) Camera Head: Each of the RGB-D cameras is mounted on a pan-tilt unit, making it possible to achieve a larger field of view around the robots. In addition to the RGB-D camera, the camera head of the autonomous robot (see Fig. 3a) is equipped with a thermal camera to detect heat emitted by victims. As thermal and RGB-D cameras are mounted close to each other, the images from both cameras can easily be aligned to detect victims more reliably (see II-B4 for details).

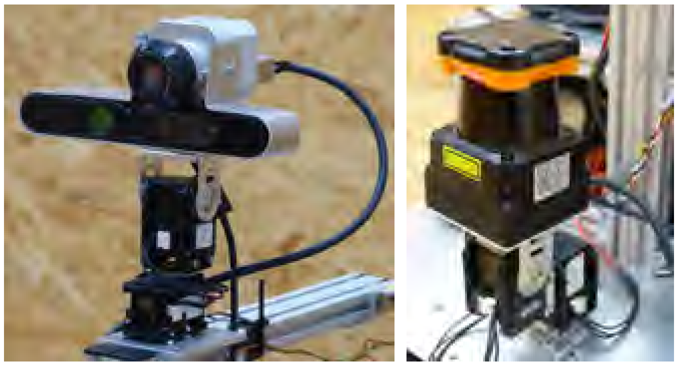

(a) (b)

Fig. 3: (a) Camera head with thermal and RGB-D camera used for victim search. (b) Gimballed laser scanner unit.

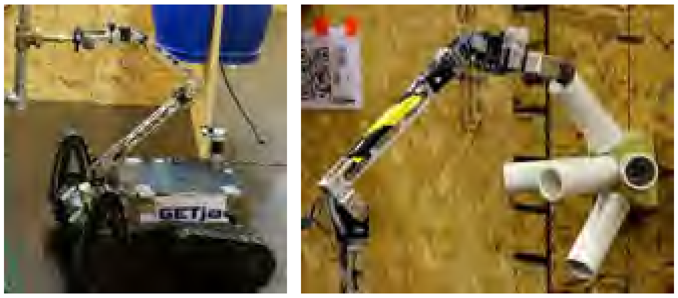

(a) (b)

Fig. 4: Self-designed robot arm with 6 degrees of freedom: (a) Arm used for turning a valve. (b) Pipestar inspection.

3) Laser Scanner Unit: The SLAM module of our mapping system is supported by a gimbaled laser scanner unit shown in Fig. 3b. This unit enables a robot to obtain high resolution 3D scans of an environment as well.

4) Robot Arm: Our teleoperated robot is equipped with a self designed robot arm (Fig. 4). The arm has 6 degrees of freedom and can be used for manipulation and inspection tasks.

B. Software

1) Control: Fig. 5 illustrates the scheme of inter-component interactions in our autonomous rescue system. The SLAM module is continuously updating the map of the environment. In parallel the exploration module uses this map to determine a path to the next optimal exploration target. The coordination between different modules is currently done using a hierarchical state machine based on SMACH [7]. This master control module requests a path from the exploration module and then forwards the way points to the navigation module one by one. In case that no path can be found, e.g. because a map is not yet available, navigation is done by choosing a suitable fallback behavior (see II-B3 for details).

While the environment is explored, the search for victims can be enabled or disabled by the master control, depending on which behavior the human operator selects before starting

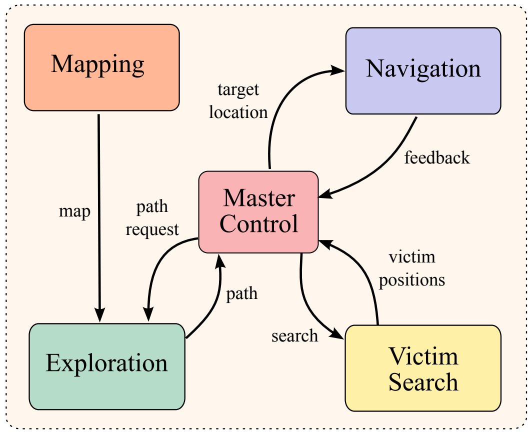


Fig. 5: Interaction of main software components (simplified).

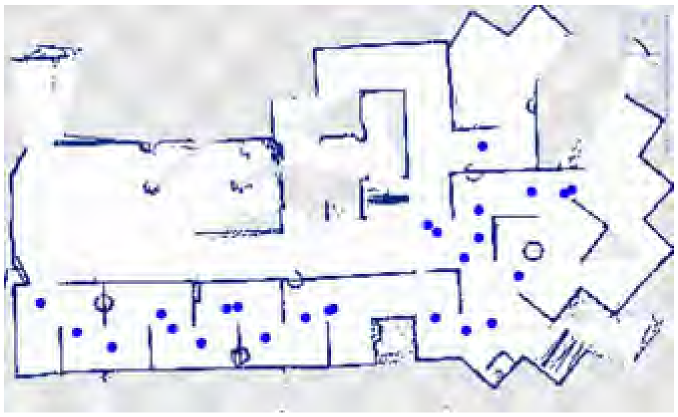


Fig. 6: Map generated by our SLAM module; detected landmarks are denoted by blue dots.

the autonomous exploration. Whenever a possible victim is detected, the position is passed to the master control in order to decide whether to continue exploration or approach the victim location for confirmation.

Modification and extension of a hierarchical state machine can be quite complex and time consuming. Therefore, we currently work on a replacement inspired by the idea of goal oriented action planning [6]. The goal is to make the decision making process flexible, scalable and easier to maintain. As a result our system will be quickly adaptable to new challenges.

2) Exploration and Mapping: Our SLAM module generates a 2D grid map by merging data coming from the robot odometry, a laser range finder and an inertial measurement unit (IMU). The SLAM core is based on the OpenKarto library [8]. A map generated by our system is shown in Fig. 6.

In order to explore an unknown environment, the map generated by the SLAM module is passed to the exploration module performing a frontier based exploration [9][10]. Here the accessibility of frontiers is first checked using the Search-Based Planning Library (SBPL) [11][12]. Afterwards we select the next exploration target based on the robot's characteristics and the costs for reaching the target, including the characteristics of the terrain (see II-B6).

Currently, we are extending our exploration and mapping modules for the case of collaborative multi robots.

3) Navigation: Usually, a real world disaster environment is partially or completely unknown and it may change over time. In order to guarantee a safe navigation in such environments, it is necessary to incorporate the sensory perceptions within the motion planning and the control loop. By this means, robots can detect environmental changes and re-plan dynamically to reach a given goal safely. For this purpose, we use a reactive obstacle avoidance navigation technique, which is developed in our lab [13]. The developed method is, in general, based on analyzing the structure of the environment and detecting free spaces (gaps) where the robot fits in. Obstacle avoidance is performed based on the configuration of obstacles between the current robot location and the selected gap, where all obstacle points are considered to compute the avoidance maneuver and simultaneously drive the robot towards the gap. This method is capable of driving a mobile robot in very dense and cluttered

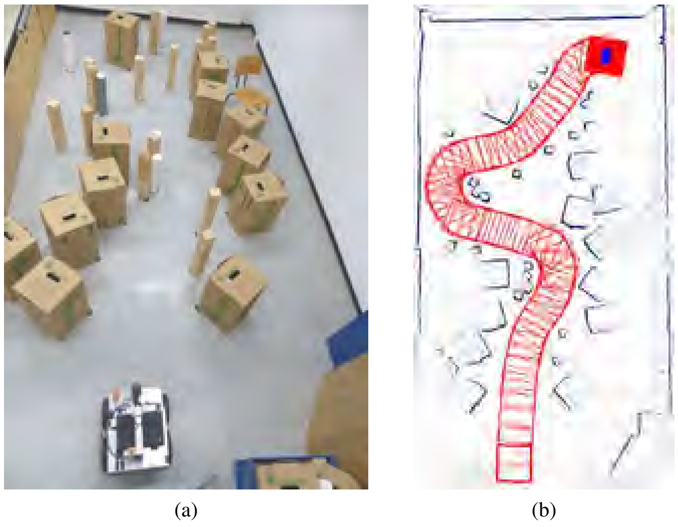
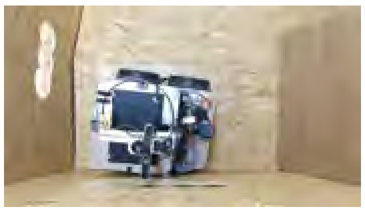


Fig. 7: An experiment carried out using only the reactive navigation module [13] in a very dense scenario; (a) Experimental setup. (b) Path followed by the navigation method.

environments with smooth and safe trajectories (see Fig. 7).

Whenever the state machine is in the exploration state, the robot has to drive towards a goal location received from the frontier based exploration module. The path planner generates the required path and the reactive navigation module is used to follow this path, generating a suitable motion command. When the robot reaches the given goal, the reactive navigation informs the path planner and waits for a new path. However, while driving towards the goal, the victim search module may announce the possibility of a victim (e.g. detecting a thermal signature of a victim). In such a case, the state machine switches over to "Target Homing" state. A variant of the reactive navigation method discussed above [14] is used to reach the assigned victim location. As soon as the robot is close enough, it stops and requests a confirmation from the operator.

In case that no path is received from the path planner (e.g. at the starting of the rescue mission where the map is not created yet), another module, wander around, is used to control the


(a)

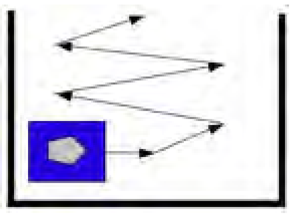

(b)

Fig. 8: Recovery behavior module: (a) Experimental setup, (b) Expected behavior.

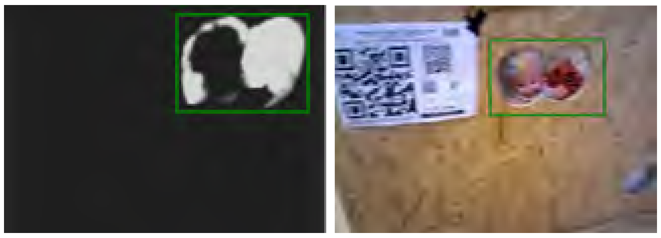


Fig. 9: Victim detection using thermal and video images.

robot. For this purposes, we use a modified version of our method proposed in [15]. The key idea here is to guide the robot towards free gaps while avoiding obstacles. A higher priority is given to gaps located in front of the robot in order to penalize moving backwards. This approach is especially suitable for maze-like environments.

In some situations the robot can get stuck (e.g. after sliding over a ramp and crashing against a wall). In such situations, a recovery behavior is activated, which tries to drive the robot towards a safe location. Once succeeded, the navigation module can take over again. Our self-developed recovery behavior drives the robot by setting several consecutive motion commands based on the robot's shape and the structure of the environment (see Fig. 8).

4) Victim Search: The main target of the robot at the RoboCup is to find simulated victims. In the last RoboCup competitions, we started to replace our old victim detection algorithm [16] with a new algorithm which is faster and more robust. The old victim detection algorithm requires more processing time to search victims, necessitating the robot to stop driving occasionally in order to process captured images. In the new algorithm, the robot searches victims while driving and detects signs of life (e.g. human body temperature, skin color, and motion). For this purpose, the robot uses a thermal camera to automatically detect human body temperature and generate a list of victim hypotheses. Consequently, visual and depth data from the RGB-D camera is used to validate the hypotheses. The validation is done by searching for skin color, holes, motion and human body features within the regions of interest identified in thermal images. To achieve this purpose, the cameras are calibrated and the images are registered (see Fig. 9). This method allows the robot to estimate the victim position and localize victims in the map.

5) QR-Code and Hazmat Symbol Detection: During the mission, the captured RGB images are used to detect QR-codes and hazmat symbols (see Fig. 10). Recently, we started to design a new system which uses a multi-camera setup for the detection of QR-codes and hazmat symbols. For the detection of hazmat symbols, we developed a method which extracts bag-of-features based on the SIFT descriptor. The classification of hazmat symbols is implemented using the support vector machine (SVM). The classifier is trained using some known hazmat symbols. For each detected object (QR-code or hazmat symbol), the 3D position is calculated and the location is plotted on a 2D map.

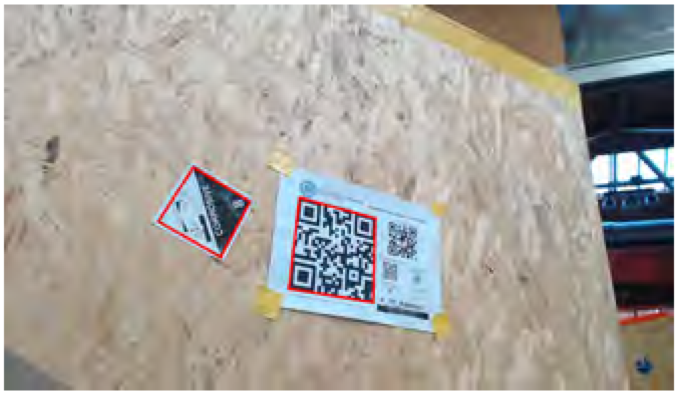


Fig. 10: QR-code and hazmat symbol detection.

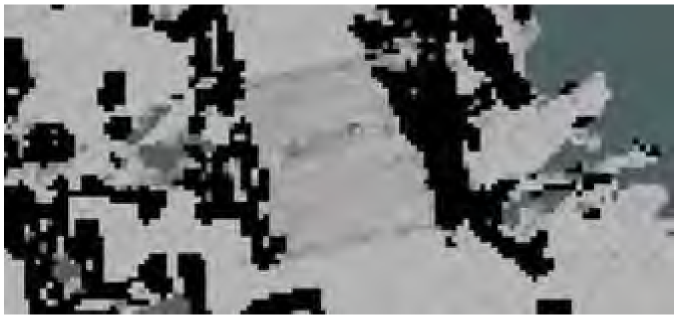


Fig. 11: Terrain map classifying the underground and marking obstacles. In the center a bidirectional ramp is classified as more difficult (darker) than flat ground (light gray).

6) 3D Processing: In the last competitions, we started processing three-dimensional data. RGB-D data is used to generate a 2.5D height map. The structure of the underground is analyzed based on the geometrical relationship of neighboring height cells to generate a trafficability map. In this way, we do not only distinguish walls and floor, but also certain difficulties for different undergrounds and detect obstacles and non-drivable passages that are not detected by the laser range finders (see Fig. 11).

Recently, we came up with a new module for 3D object recognition and tracking based on depth images. We make use of a real-time segmentation algorithm proposed in [17] to cluster a depth image and recognize objects by partial views stored in a database (see Fig. 12). This database is initially generated offline, but it can dynamically be updated online as well. The recognized objects are tracked as long as they are visible and used for further processing by other modules of our system.

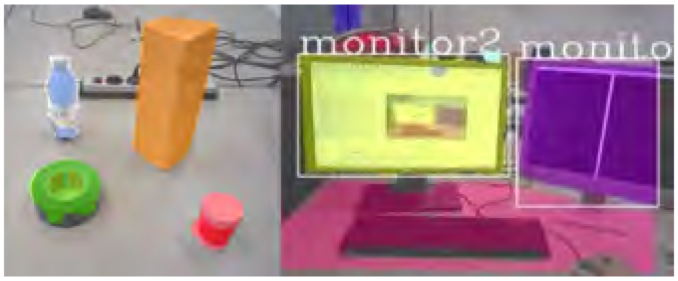


Fig. 12: Real-time segmentation for clustering a depth image (left) and object detection based on partial views (right).

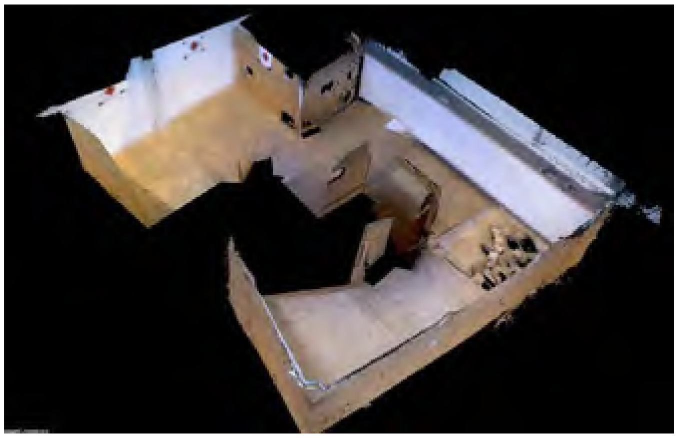


Fig. 13: Partial reconstruction of a RoboCup arena based on RGB-D data.

In the RoboCup German Open 2015, a new challenge in form of soft obstacles (vertically hanging elastic strips) was introduced. The robot had to recognize these kind of obstacles in order to drive through them. We are currently working on algorithms using depth information to detect potential soft obstacles based on depth discontinuities and edge detection. The potential soft obstacles become validated by adding a new behavior where the robot moves cautiously towards the obstacles. If the robot can move through the obstacles they are marked as soft obstacles; otherwise they will be determined as hard obstacles (see II-B1).

In addition to the 2D mapping based on laser data, a module for dense 3D reconstruction based on RGB-D data is currently under development. We make use of the InfiniTAM V2 method [18] to register depth images in real-time and adapt it to the usage for large-scale arena-like environments (see. Fig. 13). Furthermore, we are working on loop detection and closing as an extension to the algorithm. This aspect is especially difficult to solve as RoboCup arenas basically consist of wooden parts with similar textures and rectangular structures.

7) Autonomous Flipper Control: The four articulated robot arms (flippers) of our GETjag robot (see Fig. 2) are currently controlled manually by the operator because controlling the speed and direction of the robot and two pairs of flippers simultaneously needs complex and demanding control rules. Nevertheless, control by an operator raises also another difficulty: the operator perceives the environment through a camera which yields only a limited field of view. Hence, we are developing an autonomous control system to rotate the flippers according to the structure of the underground. To this end, a local height map around the robot will be taken into account based on a 3D map reconstructed over time. Up to now the autonomous control system has only been used and tested in the simulator because our 3D reconstruction module is currently under development. As soon as the 3D reconstruction is running, the control system can easily be integrated to autonomously control the flippers and disburden the operator.

C. Communication

We use wireless LAN 802.11a/b/g. Channels and power are selectable as follows:

Rescue Robot League GETbots (Germany)		
Frequency	**Channel/Band**	**Power**
5.0 GHz 802.11a	selectable	selectable
2.4 GHz 802.11b/g	selectable	selectable

D. Human-Robot Interface

A graphical user interface has been developed using the cross-platform application framework Qt [19]. To percept the status of each robot and to control the behavior, data coming from different sensors and processing units can be visualized. Furthermore, it is possible to control devices such as a robot arm remotely. The communication between robots and operator interface is based on ROS messages [20]. Fig. 14 depicts a screenshot of the interface.

The operator is able to monitor multiple robots at the same time. The number of robots is not limited and robots can be added or removed during runtime. Additionally, the operator can easily adjust the view for each robot to fit his needs. Robots in teleoperated mode are controlled using an off-the-shelf joypad.

For a fully autonomous robot, the autonomous exploration mode should be started by the operator. He can also switch the view to control a teleoperated robot. Each time an autonomous robot detects a potential victim, this incident is reported to the operator and a victim confirmation view is available to handle this event.

At the end of a mission the generated grid map containing victim and landmark positions can be saved to a storage device.

III. Application

A. Set-up and Break-Down

Our equipment includes the robots, a laptop and joystick to operate the robots, a wireless access point, and a battery backup unit to ensure autonomous electricity supply. The whole equipment is transported to the operator station on a transport cart. Our experiments have shown that setup and break-down can be easily accomplished in a couple of minutes.

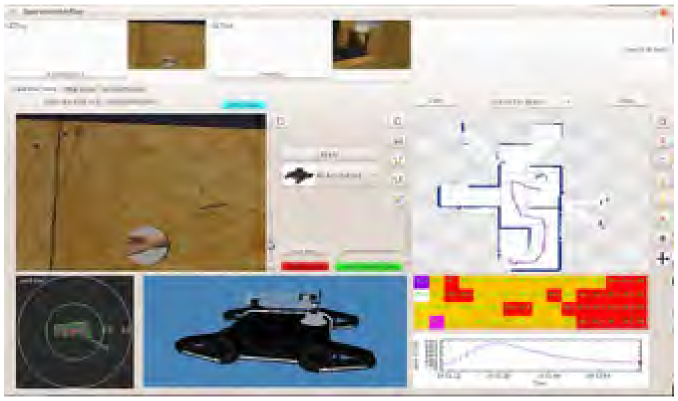


Fig. 14: Graphical operator interface.

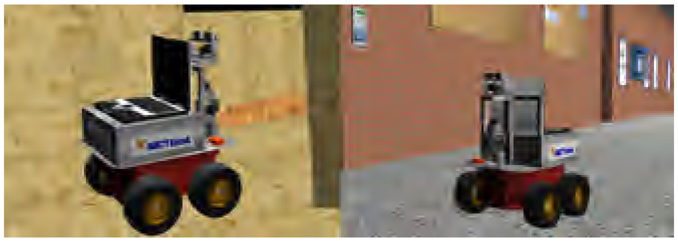


Fig. 15: SIMORE Simulation: Virtual GETbot operating in a RoboCup Rescue arena (left) and an indoor corridor (right).

B. Mission Strategy

We are going to use our two robots in parallel. While the autonomous robot is searching for victims, the operator can concentrate on exploring the orange and red parts of the arena. In this regard, he will monitor the autonomous robot and react whenever the autonomous robot detects a victim.

C. Experiments

Simulation has proven to be an auxiliary tool in the design and testing of new robot models as well as developing and evaluating a range of algorithms to run on mobile robot platforms. Especially when limitations in hardware, specific environments or time have to be considered, virtual prototyping can be a beneficial addition compared to solely testing the software in a “real” environment.

The GETbots team deploys the 3D simulation software SIMORE [3] to overcome the aforementioned limitations. Launched as a student project in our lab in 2006, SIMORE was enhanced within the following years and is now supporting the software development of our team (Fig. 15). SIMORE is based upon open source libraries such as OpenSceneGraph [21] for its rendering engine, and the Open Dynamics Engine [22] for the dynamics simulation.

It supports different types of actuators and sensors. These include e.g. pan tilt units and robotic manipulators, laser range finders, RGB-D cameras, as well as thermal cameras. ROS bindings are available also. We designed individual mazes like test arenas furnished with the typical RoboCup elements such as ramps, step fields and stairs.

For experiments with our real robots, we are using both

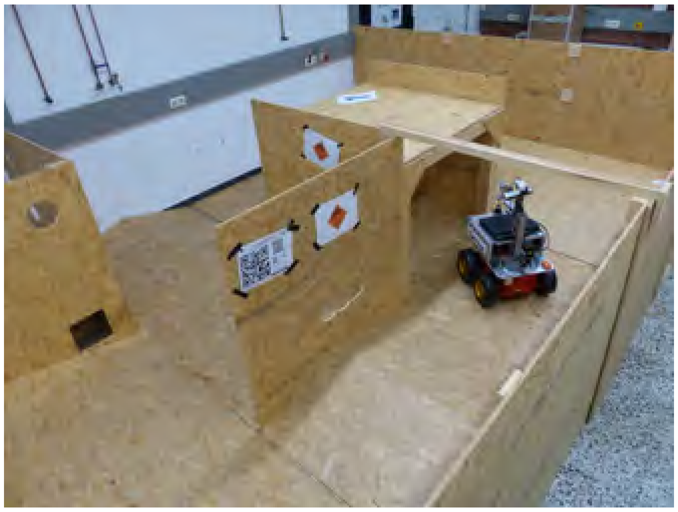


Fig. 16: Arena-like experimental setup with RoboCup elements including ramps, a stepfield and a tunnel.

individual experimental setups such as the one shown in Fig. 7a and a small arena with RoboCup elements (see Fig. 16).

D. Application in the Field

The focus of our team is developing algorithms and software which can be applied for real rescue operations. Our available platforms may not be fully appropriate for real scenarios. But we are eager to prove that the algorithms can be adapted easily to any platform which can move through disastrous environments.

IV. Conclusion

We are going to deploy two robots in the 2016 RoboCup Rescue League competition. The first robot (GETbot) works in an autonomous mode and is supposed to explore large scale arenas and detect victims. To this end, we have enhanced our previous software components and furthermore developed new modules to tackle problems for which the 3D structure of the environment has to be considered. The second robot (GETjag) is used in teleoperated mode. It is utilized to explore rough terrain, grasp objects and do inspection tasks. Since operating the GETjag is really challenging, we are developing a semi-autonomous control system to disburden the operator.

Appendix A
Team Members and Their Contributions

• Bärbel Mertsching	Management
• Dirk Fischer	System control, exploration
• Hossein Mirabdollah	Mapping and localization
• Mahmoud Mohamed	Victim search, image processing
• Muhannad Mujahed	Navigation, exploration
• Daniel Nickchen	Terrain trafficability analysis
• Mawe Sprenger	3D object recognition, operator

APPENDIX B LISTS

A. Systems List

TABLE I: Autonomous Robot

Attribute	Value
Name	GETbot
Locomotion	wheeled
System Weight	27.7 kg
Size	0.53 x 0.48 x 0.78 m
Startup time (off to full operation)	5 min
Power consumption (idle/ typical/ max)	18/ 50/ 90 W
Battery endurance (idle/ normal/ heavy load)	240/ 90/ 30 min
Maximum speed	0.7 m/s
Payload (typical, maximum)	2/ 12 kg
Support: set of bat. chargers total weight	2 kg
Support: set of bat. chargers power	115 W (230V AC)
Support: Charge time batteries (80%/ 100%)	3 h / 5 h
Support: Additional set of batteries weight	2 kg
Cost	EUR 22.530

TABLE II: Teleoperated robot

Attribute	Value
Name	GETjag
Locomotion	tracked / 4 flippers
System Weight	33 kg
Transportation size	0.6 x 0.68 x 0.38 m
Typical operation size	0.8 x 0.68 x 0.5 m
Unpack and assembly time	10 min
Startup time (off to full operation)	5 min
Power consumption (idle/ typical/ max)	33/ 75/ 142 W
Battery endurance (idle/ normal/ heavy load)	180 / 60 / 30 min
Maximum speed (flat/ outdoor/ rubble pile)	1.4/ 1/ $<$ 0.5 m/s
Carrying payload (typical/ maximum)	3/ 15 kg
Dragging payload (maximum)	50 kg
Arm: maximum operation height	95 cm
Arm: payload at full extend	$<$ 500 g
Arm: sensors	Thermopile / Camera
Telescope: height gain	15cm
Support: set of bat. chargers total weight	1 kg
Support: Charge time batteries (80%/ 100%)	3 h / 5 h
Support: Additional set of batteries weight	1 kg
Cost	EUR 17.114

TABLE III: Operator Station

Attribute	Value
Name	GETOp
System Weight	5 kg
Typical operation size	0.4 x 0.4 x 0.4 m
Unpack and assembly time	1 min
Startup time (off to full operation)	1 min
Power consumption (idle/ typical/ max)	60 / 80 / 90 W
Battery endurance (idle/ normal/ heavy load)	10 / 5 / 4 h
Power supply	APC Back UPS BE700G 700VA
WLAN router	ASUS RT-AC66U
Controller	Logitech Gamepad F3101
Cost	EUR 1.545

B. Hardware Components List

TABLE IV: Hardware Components List

Part	Brand & Model	Unit Price	Num.
Platform	Dr. Robot Jaguar-V4	EUR 13.000	1
Platform	Mobilerobots P3-AT	EUR 6.000	2
Servomotors	Dynamixel MX-106T	EUR 493	1
	Dynamixel MX-64T	EUR 299	1
	Dynamixel MX-28T	EUR 219	7
	Dynamixel AX-12A	EUR 45	3
DC/DC	TI PTN78020	USD 26	3
Batteries	Conrad Energy EC5	EUR 99	3
Batteries	Werker 12V 7.5Ah AGM	EUR 30	12
Micro controller	Arduino Pro Mini	EUR 9.50	3
Computing Unit GETjag	Mini ITX ASUS P8Z77-IDEL		
	Intel Core i5-3570T		
	8GB RAM		
	120 GB SSD	EUR 550	1
Computing Unit GETbot	ThinkPad X230	EUR 960	1
IMU	XSens MTi	EUR 1.600	1
IMU	Tinkerforge IMU Brick	EUR 60	1
Camera	iDS UI-1240ML	EUR 400	1
Camera	Logitech HD Webcam C615	EUR 65	2
Depth Camera	ASUS Xtion Pro Live	EUR 130	2
Infrared Camera	FLIR Photon 320	EUR 6.700	1
Thermopile	Melexis MLX90620	EUR 60	1
LRF	Hokuyo UTM-30LX	EUR 4.200	1
LRF	Hokuyo URG-04LX	EUR 1.600	2
CO_2 Sensor	SenseAir K30	EUR 177	1
Battery Chargers	TEnergy TB6AC	EUR 50	1
6-axis Robot Arm	Self-assembly	EUR 1.200	1
Operator Laptop	ThinkPad T430	EUR 1.300	1

C. Software List

TABLE V: Software List

Name	Version	License	Usage
Ubuntu	14.04	open	
ROS	indigo	BSD	
PCL [23]	1.8	BSD	3D object detection
ICL [17]	9.8	LGPL	3D object segmentation
InfiniTAM$^{\infty}$ v2 [18]	$>$ 2.0	ISIS	3D mapping
OpenCV [16]	2.4.8	BSD	Victim detection, Hazmat detection
slam_karto	0.7.2	LGPL	2D mapping
ZBar	0.1	LGPL	QR code detection
Operator Interface	1.2	closed source	Operator station

References

[1] "Robot Operating System," 2016. [Online]. Available: http://www.ros.org

[2] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[3] O. Kutter, C. Hilker, A. Simon, and B. Mertsching, "Modeling and Simulating Mobile Robots Environments," in *3rd International Conference on Computer Graphics Theory and Applications (GRAPP 2008)*, Funchal, Madeira, Portugal, January 2008, pp. 335 – 341.

[4] "High Performance All-Terrain Robot: Pioneer 3-AT Robot," http://www.mobilerobots.com/ResearchRobots/P3AT.aspx, 2016.

[5] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics, 2009.

[6] J. Orkin, "Three States and a Plan: The A.I. of F.E.A.R," in *Game Developers Conference*, 2006.

[7] "SMACH: Architecture for Creating Complex Robot Behaviors," 2016. [Online]. Available: http://wiki.ros.org/smach

[8] "OpenKarto GraphSLAM library," 2016. [Online]. Available: https://github.com/skasperski/OpenKarto

[9] B. Yamauchi, "A Frontier-based Approach for Autonomous Exploration," in *In Proceedings of the IEEE International Symposium on Computational Intelligence, Robotics and Automation*, 1997, pp. 146–151.

[10] L. Freda and G. Oriolo, "Frontier-Based Probabilistic Strategies for Sensor-Based Exploration," in *ICRA*, 2005, pp. 3881–3887.

[11] "Search-Based Planning Lab," 2016. [Online]. Available: http://www.sbpl.net

[12] "Search-Based Planning Library," 2016. [Online]. Available: https://github.com/sbpl/sbpl

[13] M. Mujahed, D. Fischer, and B. Mertsching, "Smooth Reactive Collision Avoidance in Difficult Environments," in *IEEE Conference on Robotics and Biomimetics (ROBIO)*, Zhuhai, China, December 2015, pp. 1471 – 1476.

[14] M. Mujahed, D. Fischer, and B. Mertsching, "Safe Gap Based (SG) Reactive Navigation for Mobile Robots," in *European Conference on Mobile Robots (ECMR)*, Barcelona, Spain, September 2013, pp. 325 – 330.

[15] M. Mujahed, H. Jaddu, D. Fischer, and B. Mertsching, "Tangential Closest Gap Based (TCG) Reactive Obstacle Avoidance Navigation for Cluttered Environments," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Linköping, Sweden, October 2013, pp. 1 – 6.

[16] Z. Aziz and B. Mertsching, "Survivor Search With Autonomous UGVs Using Multimodal Overt Attention," in *IEEE International Workshop on Safety, Security & Rescue Robotics 2010*, July 2010.

[17] A. Ückermann, C. Elbrechter, R. Haschke, and H. Ritter, "3D Scene Segmentation for Autonomous Robot Grasping," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2012, pp. 1734 – 1740.

[18] O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray, "Very High Frame Rate Volumetric Integration of Depth Images on Mobile Device," *IEEE Transactions on Visualization and Computer Graphics (Proceedings International Symposium on Mixed and Augmented Reality 2015*, vol. 22, no. 11, 2015.

[19] "Qt cross-platform application framework," 2016. [Online]. Available: http://www.qt.io

[20] "ROS Messages," 2016. [Online]. Available: http://wiki.ros.org/Messages

[21] D. Burns and R. Osfield, "Open Scene Graph A: Introduction, B: Examples and Applications," in *VR04: Proceedings of the IEEE Virtual Reality 2004*, 2004, p. 265.

[22] "Open Dynamics Engine," 2016. [Online]. Available: http://www.ode.org

[23] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.