The IUB Rugbot: an intelligent, rugged mobile robot for search and rescue operations

Andreas Birk, Kausthub Pathak, Soeren Schwertfeger and Winai Chonnaparamutt
International University Bremen (IUB)
Campus Ring 1, 28759 Bremen, Germany
a.birk@iu-bremen.de, http://robotics.iu-bremen.de/

***Abstract*— The paper describes the IUB Rugbot, a rugged mobile robot that features quite some on-board intelligence. The robot and its software is the latest development of the IUB rescue robot team, which is active since 2001 in this research area. IUB robotics takes an integrated approach to rescue robots. This means that the development of the according systems ranges from the basic mechatronics to the high-level functionalities for intelligent behavior.**

FINAL VERSION

```
@incollection{
rugbot_ssrr06,
Author = {Birk, Andreas and Pathak, Kausthub
     and Schwertfeger, Soeren and
     Chonnaparamutt, Winai},
Title = {{The IUB Rugbot: an intelligent,
     rugged mobile robot for search and
     rescue operations}},
BookTitle = {IEEE International Workshop on
     Safety, Security, and Rescue Robotics
     (SSRR)},
Publisher = {IEEE Press},
Year = {2006} }
```

I. INTRODUCTION

Rescue robotics features an interesting combination of allowing basic research while being application oriented [1]. The focus for the application oriented systems is on rugged, simple platforms that mainly act as mobile cameras [2][3][4]. On the other hand, any bit of intelligence in the sense of perception and world-modeling capabilities up to autonomy is very useful not only from a basic research but also from an application perspective. Researchers in the rescue community accordingly work on issues like mapping or fully autonomous victim detection [5] and navigation as well as exploration [6], [7], [8]. The work on the so to say intelligence side of the robots even ranges up to multi robot teams [9] [10]. But advanced locomotion systems are also a topic of research in the rescue community. Examples include jumping [11] and snake robots [12]. Last but not least, the human machine interaction as well as the operator station itself are of scientific interest [17].

The IUB rescue robot team is interested in the basic research aspects of the field while at the same time trying to develop fieldable solutions. IUB robotics is engaged in this research area since 2001 and it has participated in many RoboCup rescue robot competitions that gave valuable feedback about the performance of the systems [13] [14] [15] [16]. The latest development of the group is the so-called Rugbot (figure 1) that provides a sturdy, rugged platform suitable for real applications while providing significant computation power and sensor interfaces to allow basic research on the many open questions in this field and robotics in general.

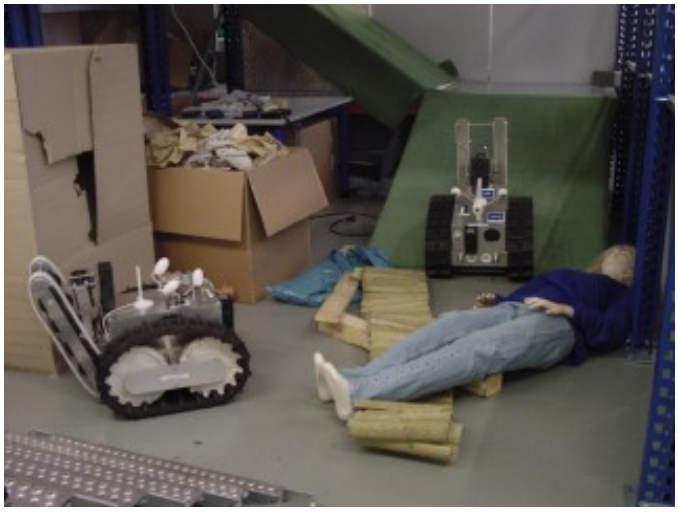

Fig. 1. Two Rugbots in the IUB rescue arena.

The rest of this paper is structured as follows. Section two gives an overview of the hardware of the Rugbots. In section three the basic software aspects are presented. Section three gives a more in detail description of the higher level software onboard the robots. The software on the operator station is presented in section four. Section five concludes the paper.

II. THE ROBOT HARDWARE

As mentioned in the introduction, IUB robotics takes an integrated approach to rescue robotics. The group develops the system from scratch, ranging from the electronics over the mechanics up to the higher software levels. The Rugbot robot type (figure 1) is the latest result of these efforts. The early versions of the Rugbot were used in the RoboCup 2005 competition in Osaka [13]. The Rugbot is based on the so-called CubeSystem, which is a collection of hardware and software components for fast robot prototyping [18], [19]. The CubeSystem consists of the RoboCube controller hardware [20], a special operating system called CubeOS [21] and libraries for common robotics tasks [22].

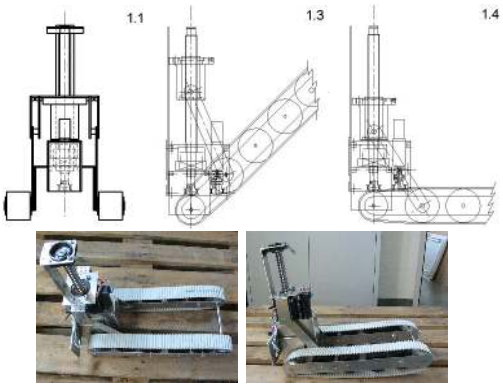

Fig. 2. The support flipper of Rugbot is based on a special ball screw design that can take large forces.

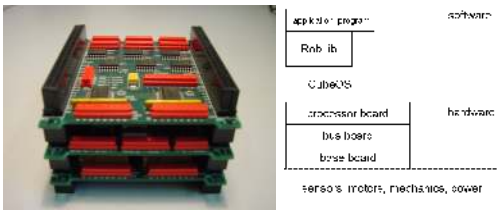

Fig. 3. On the left, a RoboCube processor-, bus- and I/O-board stacked together, leading to a compact controller hardware. On the right, the structure of a CubeSystem application including hard- and software.

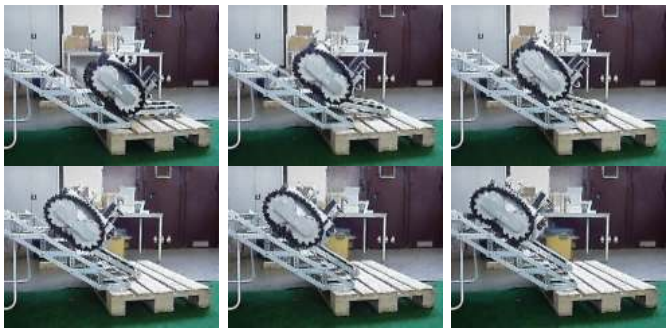

Fig. 4. Rugbot going up stairs.

The Rugbots are tracked vehicles. They are relatively lightweight (about 35 kg) and have a small footprint (approximately 50 cm x 50 cm). They are very agile and fast in unstructured environments and they also perform well on open terrain. This holds for tracked vehicles in general [23][24], which can also be seen by their popularity in the RoboCup rescue league, for example in the robots of Team Freiburg, Robhaz, Casualty or IRL [25] [26] [27] [28]. A special feature of rugbot is an active flipper mechanism (figure 2) that allows to negotiate rubble piles and stairs (figure 4). Rugbots have significant computation power in form of an onboard PC and they can be equipped with a large variety of sensors. The standard payload includes a laser-scanner, ultrasound sensors, four cameras, one thermo camera, and rate gyros. The onboard software is capable of mapping, detection of humans and fully autonomous control, teleoperation is in varying degrees is also supported [29]. The on-board batteries allow for 2.5 to 3 hours of continuous operation including moving through rough terrain.

III. Overview of the Rugbot's intelligent software

The intelligent software on the Rugbots is designed to support the whole range from teleoperation to full autonomy. Its lowest layer resides on the RoboCube where a hard realtime system provides all functionalities for the basic motor and motion control [29]. The so to say higher intelligence functionalities are implemented on an additional high-performance PC on the robot. Finally, an operator station in form of an additional PC is used to provide the essential data to a human operator. Nevertheless, it is important to note that the operator station purely serves this purpose and that all crucial software for the run to run is implemented on board the robot itself.

In addition to the core higher level software on the robot as well as on the operator station, which is describe in more detail in the following sections, the group engages in several basic research activities. These include work on a special GUI for adjustable autonomy [30], mapping of large areas with multiple robots [31] [32], and exploration under the constraints of wireless networking [7] [33].

IV. The on-board software

A. Robot Server

The *robot-server* is a multi-threaded program running at about 10 Hz. on a Linux computer mounted on the robot. It has been coded in C++. All system-wide constants like port numbers, resolutions, etc., are read at startup from a configuration file. The various tasks performed by this process are:

- *Operator GUI interaction:* The NIST RCS framework [34] has been used to handle communication between the robot-server and the operator GUI. This framework allows data to be transferred between processes running on the same or different machines using Neutral Message Language (NML) memory buffers. All buffers are located on the robot computer, and can be accessed by the operator GUI process asynchronously. A schematic of the various buffers used is shown in Fig. 6.
- *Communication with the robot drive:* As mentioned before, the IUB robotics group has developed a collection of hardware and software components for fast robot prototyping called the *CubeSystem*. The three main parts of this system are:
  1) *RoboCube*: This is an embedded controller family based on the MC68332 processor.
  2) *CubeOS*: An operating system family.
  3) *RobLib*: a library with common functions for robotics. These functions include motor speed commands, odometry computations, and control of accessory hardware like flipper and lights.

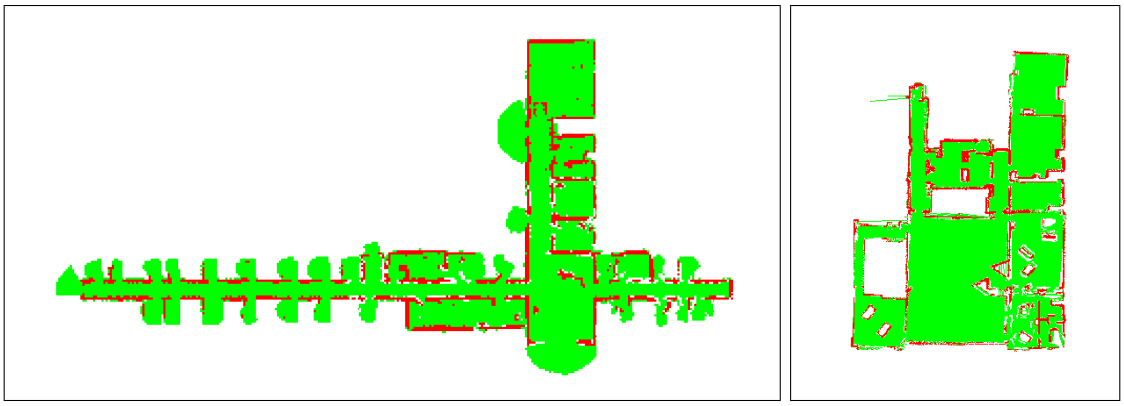

Fig. 5. Two maps of large buildings generated by multiple robots. On the left, a map from 6 robots running in simulation. On the right, a result based on the real world data of 4 robots.

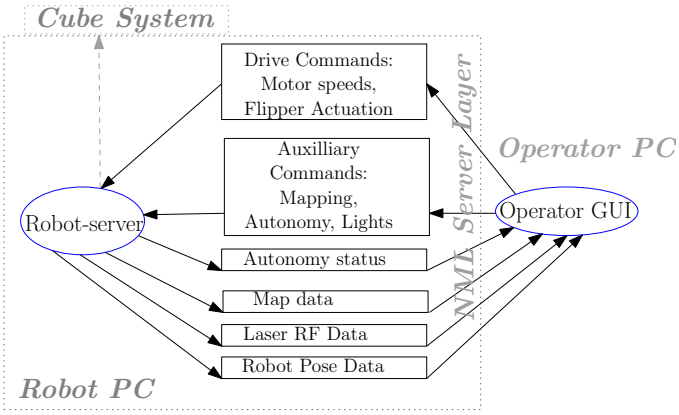

Fig. 6. Communication between the robot-server and operator-GUI using NML buffers.

The robot-server uses serial communication to transfer operator drive commands from the NML buffer to the Cube system.

- *Sampling sensors:* The IUB Rugbot is equipped with several onboard sensors. Some of these are briefly described in the following:
  1) *Laser Range Finder:* A HokuyoURG04-LX has been used [36]. It has an angular field of view of 240^o and angular resolution of 0.36^o. The range accuracy is ± 10 mm.
  2) *Gyro:* An XSense MTi 3DOF gyro [37] as been used. It has a static accuracy of 0.5^o in roll/pitch and 1^o in heading direction. The robot-server uses the gyro in polling mode to recalibrate the Cube system's odometry heading direction. This technique enables the creation of satisfactory maps using odometry alone, without the use of scan-matching. Please also refer to the description of mapping below.
- *Mapping:* The robot-server process can run several mapping algorithms in separate threads for comparison. Whenever new pose and LRF data becomes available, all the mapping threads are notified. The map data is transferred via NML buffers to the operator GUI, which can switch between available maps. Currently two mapping algorithms have been tested:
  - A basic occupancy-grid ray-tracing algorithm which uses LRF and odometry data. The odometry heading direction is corrected using the gyro data as mentioned earlier.
  - A Grid based particle filter SLAM algorithm developed by Grisetti et al [39].

B. Cameras

Several static webcams are mounted on the robot to give the operator a good view of the robot's surroundings. The streaming of camera data from the robot to the GUI is not handled inside the robot-server using NML. Instead, separate palantir [38] server processes are spawned for each static webcam.

Additional fexibility is provided by a centrally mounted Panasonic KX-HCM280 pan-tilt webcam with optical zoom. The control of this camera is done directly by the operator GUI, using an *http* based protocol.

V. The Operator Station Software

The operator hardware consists of a PC or a Laptop running Linux. Additionally a Gamepad is attached to the computer via USB. The main operator software has a graphical user interface to provide information and interact with the operator. Alternatively a text-only operator may be used.

A. Design Decisions

The GUI consist of a single window that is using Qts Grid Layout. This way the contents scale with the window size and the GUI is thus optimally usable with different resolutions. Former operator implementations were using multiple windows for video streaming, map display and others. This turned out to be difficult to use and quite error-prone.

The design of the GUI itself was inspired by the idea, that the information needed for the actual driving of the robot should be on the top of the window while all additional information should be in the lower parts. Additional design

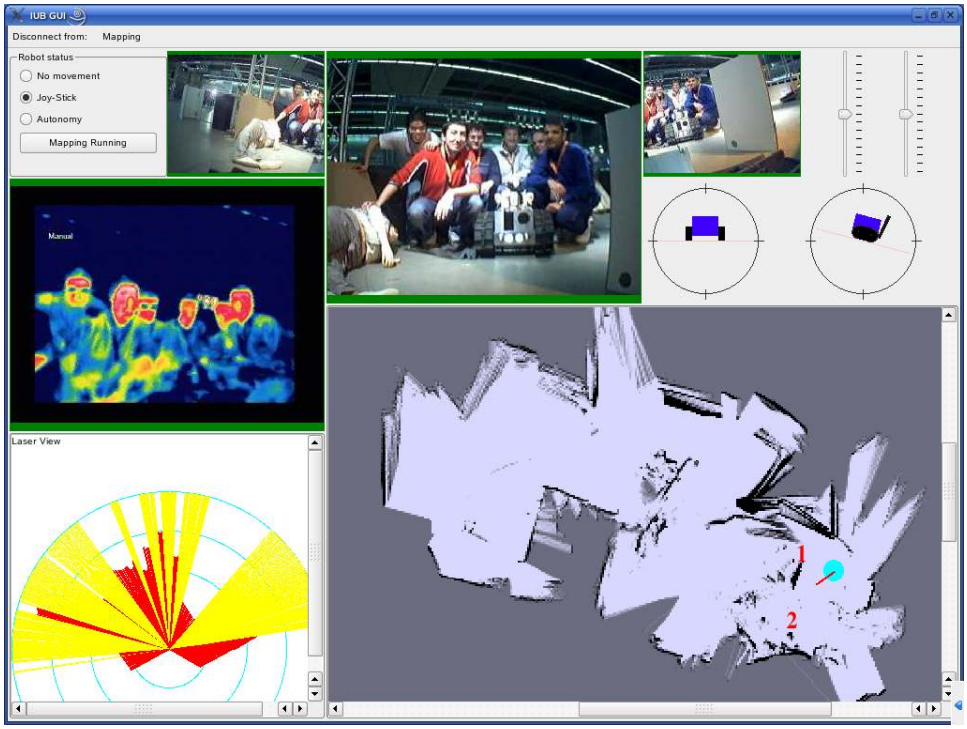

Fig. 7. The standard GUI for controlling an IUB rescue robot.

decisions had to be made since the available space inside the window became short quite fast. Controls that are used quite rarely (like ”connect to”) went into the menu and a tab was introduced to hold the map, a big video display and the log.

B. Main control

The most important controls are located in the attention-heavy top left corner of the window. Here the user can change the movement status of the robot by choosing between ”no movement”, ”joystick movement” and ”autonomy”. The option to (temporally) switch off the mapping when driving in heavy terrain and the light switch are located underneath this control.

C. Driving information

The driving information are presented at the top of the window. In the center the streamed video from the front camera is displayed with a resolution of 320 times 240 pixel. The view of the left and the right track which are usefull to circumscribe obstacles are displayed on the left respectively right side of the front view with a resolution of 160 times 120 pixel. A modified palantir client is used for this video streaming which supports stretching over the streams actual resolution. This client also has a context menu which allows to disconnect, reconnect or connect to a different server.

In the top right corner different status information of the robot are displayed. The joystick position determining the speed is shown for each track as well as a bar indicating the resistance the motors have to overcome. This is usefull for stall detection. Two displays for the roll and the pitch are shown below those bars. The roll is indicated in the left one while the pitch of the robot is represented on the right. Additionally the current position of the flipper is displayed, taking the pitch into account which allows the user to easily estimate the flipper angle compared to flat ground.

D. Secondary information

In the bottom part of the window information that are of less importance during driving the robot are displayed. On the left side the latest laser range finder scan and the infrared camera are shown. The infrared camera uses the same palantir client as the front and track views. It is used to find hidden victims and to get more information about a victims status. The laser scanner canvas is usefull to find out whether the robot will fit through a passage or not and to get a feeling about the surrounding of the robot since it has an opening angle of 280 degrees.

The biggest widget in the GUI is the tab located on the bottom left. The tab is used to display detailed information with great resolution which is, on the other hand, not used that often. The following tabs are used or in development.

- **T**he Panasonic viewer Some robot have a Panasonic KX-HCM280 webcam mounted. This is a pan-tilt camera which supports a real zoom. A client software running in its own thread has been written to control the camera and to receive and paint the pictures. The camera is mainly controlled via the gamepad which is described later.
- **T**he log Here all messages coming from the robot, the operator software or the user are displayed. There are three different priority levels: Error messages, warnings and notices. The purelylog is filterable by topics and

sortable by each row. The time that elapsed between two log entries is displayed when marking these entries.

- **T**he maps The maps are generated and stored in the robot. Every two seconds the latest map is transferred to the operator using the NML buffer. This update is indicated by a slight colorchange of the map background. The map is zoomable (with the + and - keys) and scrollable. A right click on the maps creates a new entry in the victim list. The position of the victim can be corrected with drag and dropping the number on the map. A victim dialog will pop up when doubleclicking on the victim number.
  Currently two different maps are shown. The first map was generated using the odometry information which were corrected using the gyro. The second map uses the Grisetti algorithm. A third map is being developed that implements scan matching.
- Another tab that is being developed is used to display three dimensional sensor data gathered by a 3D laser scanner.

E. The victim dialog box

In this dialog box the user can fill in information that he gathered for victims. After doubleclicking on the victim number the following information about the the victim can be filled in: tag, form, motion, Heat, sound, Co2, state (an editable pull down box), situation (an editable pull down box) and annotations. The dialog shows the map and has two checkboxes. If the "Save Image" checkbox is activated, the current images of the front view and the thermo camera are saved (or updated) to include them in the report. The "Bug me again" checkbox is currently not used. There are three options to save the dialog box. "Confirm" saves the victim, "reject" deletes it and "unsure" doesn't override the victim report files but keeps the current information.

F. The victim report

The victim report is generated using static tex files together with automaticly generated ones. It features an overall map. The user has to generate this overview map whenever he thinks he has a good map. For that he has to go to the GUI menu, select mapping "Mapping", and press "Save Map". The software then automatically rotates the map to the north and saves it as a png file in the report directory. Once a victim is confirmed in the victim dialog a tex file is written containing all the information the user provided. This way no victim is lost if the GUI should happen to crash. A script in the report folder automatically generates the pdf report using pdflatex and opens it with acroread.

G. Steering the robot

A game pad is used to steer the robot. For that reason the game pad thread is only started when a connection to a robot is active. The analog joystick is used for smoothly driving the robot. For that values for the x and y axis are converted into speed commands for the left and right tracks. The slider is used to lower the maximum speed thus enabling precise movement in narrow environments.

The gamepad features several buttons. The two right most buttons (Z and C) are used for the flipper movement. The upper button (Z) moves the flipper up and the lower button (C) moves it down. The S button is used to stop the flipper. There are two buttons on the back of the joystick. The right one is used to switch between the tabs in the GUI. This is a convenient way of switching between the map tabs for orientation and the Panasonic camera tab when searching for victims. The other buttons are for the control of the Panasonic pan tilt zoom camera. Pressing left back button while moving the joystick pans and tilts the camera. The two middle buttons (Y and B) are for zooming in and out. The Top left button (X) homes the camera, which means that the camera will look forward and the standard zoom will be applied.

H. GUI implementation

The GUI uses Trolltechs cross-platform Qt Class Libraries. Several threads are running in parallel to handle the massive amount of data. Each camera view has its own thread (we use a modified palantir client to lower refresh rate which saves bandwith. Palantir uses socket connections. The Panasonic webcam view also runs in its own thread using the cameras HTTP protocol. All other data is send via the nml using the nml data buffers. The buffers are read within an own thread and mutexes and signals are used to process the data in other threads. The joystick is, last but not least, controlled by a separate thread, too.

I. Text-mode operator

The GUI inherits its basic functions from an operator class which only has text output. It is thus possible to steer the robot using the joystick without any GUI. In this case the camera images could be displayed (supposedly also on another computer) using the palantir client or a webbrowser for the Panasonic webcam.

VI. Conclusion

The IUB Rugbot was introduced in this paper. It is the latest development from IUB robotics in the field of rescue robotics. The robot is very rugged and mobile as it is intended to support fieldable solutions. At the same time it offers substantial processing power and on-board sensors. This allows the development of advanced higher level software as presented in this paper.

References

[1] A. Birk and S. Carpin, "Rescue robotics - a crucial milestone on the road to autonomous systems," *Advanced Robotics Journal*, vol. 20, no. 5, 2006.

[2] M. M. R. Murphy, J. Casper and J. Hyams, "Potential tasks and research issues for mobile robots in robocup rescue," in *RoboCup-2000: Robot Soccer World Cup IV*, ser. Lecture notes in Artificial Intelligence 2019, T. B. Peter Stone and G. Kraetszchmar, Eds. Springer Verlag, 2001.

[3] R. G. Snyder, "Robots assist in search and rescue efforts at wtc," *IEEE Robotics and Automation Magazine*, vol. 8, no. 4, pp. 26–28, 2001.

[4] A. Davids, "Urban search and rescue robots: from tragedy to technology," *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, vol. 17, no. 2, pp. 81–83, 2002, tY - JOUR.

[5] S. Bahadori, L. Iocchi, D. Nardi, and G. Settembre, "Stereo vision based human body detection from a localized mobile robot," in *Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.*, 2005, pp. 499–504, tY - CONF.

[6] D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi, "Autonomous navigation and exploration in a rescue environment," in *Safety, Security and Rescue Robotics, Workshop, 2005 IEEE International*, 2005, pp. 54–59, tY - CONF.

[7] M. Rooker and A. Birk, "Communicative exploration with robot packs," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence (LNAI), I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, Eds. Springer, 2006.

[8] M. N. Rooker and A. Birk, "Combining Exploration and Ad-Hoc Networking in RoboCup Rescue," in *RoboCup 2004: Robot Soccer World Cup VIII*, ser. Lecture Notes in Computer Science, D. Nardi, M. Riedmiller, and C. S. e. al., Eds. Springer-Verlag GmbH, 2005.

[9] A. Farinelli, G. Grisetti, L. Iocchi, S. Lo Cascio, and D. Nardi, "Design and evaluation of multi agent systems for rescue operations," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 4, 2003, pp. 3138–3143 vol.3, tY - CONF.

[10] A. Farinelli, L. Iocchi, D. Nardi, and V. Ziparo, "Task assignment with dynamic perception and constrained tasks in a multi-robot system," in *Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 1523–1528, tY - CONF.

[11] H. Tsukagoshi, M. Sasaki, A. Kitagawa, and T. Tanaka, "Design of a higher jumping rescue robot with the optimized pneumatic drive," in *Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 1276–1283, tY - CONF.

[12] T. Kamegawa, T. Yamasaki, and F. Matsuno, "Evaluation of snake-like rescue robot "kohga" for usability of remote control," in *Safety, Security and Rescue Robotics, Workshop, 2005 IEEE International*, 2005, pp. 25–30, tY - CONF.

[13] A. Birk, S. Carpin, W. Chonnaparamutt, V. Jucikas, H. Bastani, I. Delchev, I. Krivulev, S. Lee, S. Markov, and A. Pfeil, "The IUB 2005 rescue robot team," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence (LNAI), I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, Eds. Springer, 2006.

[14] A. Birk, "The IUB 2004 rescue robot team," in *RoboCup 2004: Robot Soccer World Cup VIII*, ser. Lecture Notes in Artificial Intelligence (LNAI), D. Nardi, M. Riedmiller, and C. Sammut, Eds. Springer, 2005, vol. 3276.

[15] A. Birk, S. Carpin, and H. Kenn, "The IUB 2003 rescue robot team," in *RoboCup 2003: Robot Soccer World Cup VII*, ser. Lecture Notes in Artificial Intelligence (LNAI), D. Polani, B. Browning, A. Bonarini, and K. Yoshida, Eds. Springer, 2004, vol. 3020.

[16] A. Birk, H. Kenn, M. Rooker, A. Akhil, B. H. Vlad, B. Nina, B.-S. Christoph, D. Vinod, E. Dumitru, H. Ioan, J. Aakash, J. Premvir, L. Benjamin, and L. Ge, "The IUB 2002 rescue robot team," in *RoboCup-02: Robot Soccer World Cup VI*, ser. LNAI, G. Kaminka, P. U. Lima, and R. Rojas, Eds. Springer, 2002.

[17] T. Kimura and M. Ishizaki, "Development of an operating board for rescue robots considering safety and misuse of operators," in *Safety, Security and Rescue Robotics, Workshop, 2005 IEEE International*, 2005, pp. 66–68, tY - CONF.

[18] A. Birk, "Fast robot prototyping with the CubeSystem," in *Proceedings of the International Conference on Robotics and Automation, ICRA'2004*. IEEE Press, 2004.

[19] A. Birk, H. Kenn, and T. Walle, "On-board control in the RoboCup small robots league," *Advanced Robotics Journal*, vol. 14, no. 1, pp. 27 – 36, 2000.

[20] ——, "RoboCube: an "universal" "special-purpose" hardware for the RoboCup small robots league," in *4th International Symposium on Distributed Autonomous Robotic Systems*. Springer, 1998.

[21] H. Kenn, *CubeOS, The Manual*. Vrije Universiteit Brussel, AI-Laboratory, 2000.

[22] A. Birk, H. Kenn, and L. Steels, "Programming with behavior processes," *International Journal of Robotics and Autonomous Systems*, vol. 39, pp. 115–127, 2002.

[23] F. Hardarsson, "Locomotion for difficult terrain," Mechatronics Lab, Dept. of Machine Design, Tech. Rep., 1997.

[24] J. Y. Wong, *Theory of Ground Vehicle, 3rd edition*. John Wiley and Sons, Inc., 2001, ch. 4.5.

[25] A. Kleiner, B. Steder, C. Dornhege, D. Meyer-Delius, J. Prediger, J. Stueckler, K. Glogowski, M. Thurner, M. Luber, M. Schnell, R. Kuemmerle, T. Burk, T. Bräuer, and B. Nebel, "Robocuprescue - robot league team rescuerobots freiburg (germany)," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence (LNAI), I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, Eds. Springer, 2006.

[26] W. Lee, S. Kang, S. Lee, and C. Park, "Robocuprescue - robot league team ROBHAZ-DT3 (south korea)," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence (LNAI), I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, Eds. Springer, 2006.

[27] M. W. Kadous, S. Kodagoda, J. Paxman, M. Ryan, C. Sammut, R. Sheh, J. V. Miro, and J. Zaitseff, "Robocuprescue - robot league team CASualty (australia)," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence (LNAI), I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, Eds. Springer, 2006.

[28] T. Tsubouchi and A. Tanaka, "Robocuprescue - robot league team Intelligent Robot Laboratory (japan)," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence (LNAI), I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, Eds. Springer, 2006.

[29] A. Birk and H. Kenn, "A control architecture for a rescue robot ensuring safe semi-autonomous operation," in *RoboCup-02: Robot Soccer World Cup VI*, ser. LNAI, G. Kaminka, P. U. Lima, and R. Rojas, Eds. Springer, 2003, vol. 2752, pp. 254–262.

[30] A. Birk and M. Pfingsthorn, "A hmi supporting adjustable autonomy of rescue robots," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence (LNAI), I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, Eds. Springer, 2006.

[31] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," *IEEE Proceedings, special issue on Multi-Robot Systems*, accepted.

[32] S. Carpin, A. Birk, and V. Jucikas, "On map merging," *International Journal of Robotics and Autonomous Systems*, vol. 53, pp. 1–14, 2005.

[33] M. Rooker and A. Birk, "Combining exploration and ad-hoc networking in robocup rescue," in *RoboCup 2004: Robot Soccer World Cup VIII*, ser. Lecture Notes in Artificial Intelligence (LNAI), D. Nardi, M. Riedmiller, and C. Sammut, Eds. Springer, 2005, vol. 3276, pp. pp.236–246.

[34] The NIST Real-Time Control Systems Library. http://www.isd.mel.nist.gov/projects/rcslib/, 2006.

[35] A. Birk, The Cube System. IUB Robotics Website. http://robotics.iu-bremen.de/CubeSystem/, 2006.

[36] Hokuyo Scanning Laser Range Finder. http://www.hokuyo-aut.jp/products/urg/urg.htm, 2006.

[37] XSens MTi 3 DOF Gyro. http://xsens.com, 2006.

[38] Palantir: A multichannel interactive streaming solution. http://www.fastpath.it/products/palantir, 2006.

[39] G. Grisetti, C. Stachniss, and W. Burgard. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2005. http://www.informatik.uni-freiburg.de/ stachnis/research/rbpfmapper