RoboCup Rescue 2016 Team Description Paper Rescube

Péter Kopiás and Márton Krauter

Info

Team Name:	Rescube
Team Institution:	Datastart Kft.
Team Leader:	Péter Kopiás
Team URL:	http://rescube.hu/

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

***Abstract*—The Team Description Paper of team Rescube gives a comprehensive overview of the team background, its robots and infrastructure in development for the 2016 RoboCup Championship in Leipzig. Based on the experiences of the last competition, team Rescube has designed a versatile four wheel driven robot with a robotic arm with the ability to effectively meet the skills required at Rescue competitions. It features a special wheel design with variable geometry and an extensible "giraffe-neck" that affords it great flexibility.**

***Index Terms*—RoboCup Rescue, Team Description Paper, Rescube.**

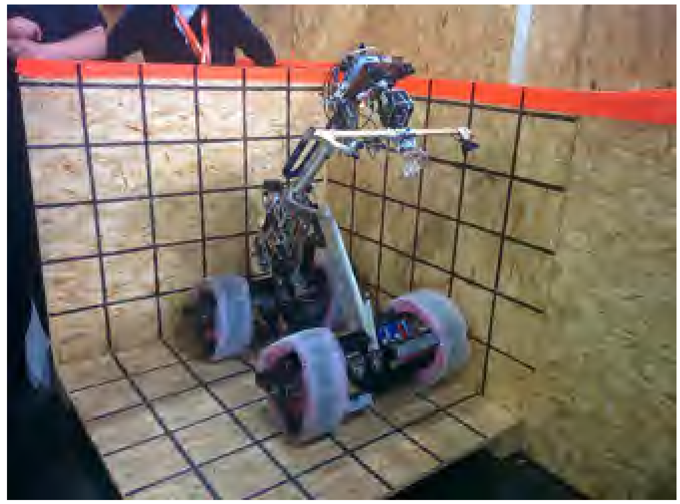

Fig. 1. Barrel2015 at the RoboCup German Open (20cm gridsize)

I. Introduction

THE members of team Rescube are young robotics enthusiasts from Hungary who share a challenge seeking attitude and willingness to hone their problem solving skills in common. With roots dating back to the 2000s the idea of Team Rescube was ignited by the RoboCup World Cup 2013 event and the team made its debut at the 2015 RoboCup German Open.

A. Team Background

While most team members are mechanical, electrical or software engineers we also have numerous university students of diverse areas. We are self motivated and run the project by solely relying on personal and sponsorship budget in parallel to maintaining strong partnership with top Hungarian scientific and technology universities.

Our efforts in RoboCup Rescue Robot League are pioneering to raise awareness of the Hungarian academic and industrial sector to the matters of the young engineers. Besides trying to establish an environment where the engineering students could work on interesting challenges, our intention is to encourage the next generation of students in choosing scientific and engineering fields. We truly believe that with continuous efforts we could create a great balance of tasks, knowledge and resources to let the creative ideas grow into solutions. We volunteer to help in organizing Hungarian robotics competitions, and to hold presentations to motivate the youngest to work hard and achieve their goals.

We strive to maintain an open community characterized by tolerance, respect for the individual and a minimum of hierarchy. Last but not least we believe having fun together plays an indispensable role in any successful teamwork.

B. Improvements over Previous Contributions

1) Team conclusions: As this is a non-profit, voluntary hobby of the members, the resources available depend on members personal background and obviously on their morale. After the German Open 2015, the team members were interviewed to gather and summarize the lessons learned and the take-aways to synthesize actions to be taken for a more successful next championship.

The key areas revealed (but are not limited to):

- importance/lack of practice and time management
- clear definition and communication of who-does-what-when
- improving handling of inevitable team fluctuation (knowledge transfer)
- lack of expertise in specific practice areas
- suboptimal logistics

We are actively seeking for the opportunities to improve and to widen the major and minor bottlenecks identified.

2) Software issues: Although using ROS as a software framework is an obvious choice, its learning curve (respectively that of the myriad possibly useful packages of varying

Péter Kopiás is with the University of Óbuda, Budapest, Hungary, e-mail: kopias.peter@gmail.com.

Fig. 2. Shape shifting in action

level of documentation) is steep, and we had to distinguish between the software modules we spend time/energy on. Based on the experience of the last competition it was clear that a unified software development environment is vital for the effective teamwork. This year we have put serious efforts to create a virtual development environment (using Vagrant and Ansible for configuration management) available to all team members at any time. As a result now we can have multiple virtual machines servers (also for the operator and robots computers) while all the configuration data is declarative and are contained in a version control system (git). The team-wide use of this development environment enables the new members to start contributing faster, use machines of each other's, and also helps keeping the software stack synchronised and centralises the management tasks to the senior members hands thus reducing maintenance costs. We have made plans to release the devops toolchain to the community, when it reaches the appropriate level of maturity.

In 2015, as a newcomer team we were focusing on teleop mode, now we are developing our semi-autonomous systems, starting with sensor processing, decision making and intuitive user interfaces. 3D object recognition, visual image processing, navigation and task repetition are on the backlog, and we have the GNSS-based outdoor competition set as a goal as well.

3) Networking issues: We have had some wireless connection problems last year, as the network hardware was limited to some congested channels. Now we are using a more robust, more configurable networking infrastructure.

4) Engineering improvements: In 2015 our robot had a variable geometry construction, with the ability to change its wheelbase and height (Fig. 2). This also gave the robot a simple arm like structure. Although this system worked very well in the different scenarios (Fig. 3), we learned that in the special case of tight turns on ramps it is very hard to control the robot precisely. As the variable geometry is a great feature when moving on rough terrain, we have enhanced the concept to make it more compact while still being able to climb extreme ramps and stairs. The solution is to use a linear actuator as the main backbone, and all the sensors, motors, and the robotic arm is built around that.

Also while the previous "giraffe-like" robot worked just fine, we see a lot of room for improvements and optimizations of its the features by building a more capable robotic arm with a much lighter sensor head. Thus we could still open doors and pick up objects, but at the same time the robot would get higher degrees of freedom, and could transfer the center of weight in a more precisely controlled manner when needed in critical situations (Fig. 4).

Fig. 3. Variable geometry allows the robot to navigate the sensors near this victim

Fig. 4. Climbing a steep ramp

II. System Description

Our credo is that robots should serve a purpose. The RoboCup Rescue competition defines the scope we are focusing on, and with every development iteration we use the gathered information to optimize the current configuration. Of course there are some general key decisions we had to make ahead, which must remain the same throughout the development. The primary guidelines and decisions are:

- rigid body needs more weight thus forces the powertrain to scale up, making the robot expensive and failure prone (as tracks require a rigid body, we have opted for elastic chassis with all wheel drive and wheels optimized for the task)
- flexibility, torsion and deformation should be a design goal. The robot might lose some features while on mission, but must never stop moving. The planned sacrifice of replaceable parts could save the day in a competition or save a life in real disaster situations.

- minimizing the static surface under the robot (if only moving parts could ever touch the terrain, the only constraints of movement are the traction, center of gravity and motor power)
- dynamic wheel configuration (that way the robot would always have a degree of freedom to move, even when some wheels do not have traction)
- keep everything simple. We are working hard to never add parts without a good reason and we are trying to keep the system complexity at the minimum
- there is a defined powertrain cascade: a planned series of graceful degradation in the wheel-gear-motor-esc-battery line. When stuck it is acceptable to lose a part from the wheels, or even lose a whole wheel, but it is not acceptable for a motor controller to fail at the maximum rated power consumption of the motors. So each part in the chain is more durable than the previous item, and this way the robot could still operate with reduced features
- every part must be accessible and repairable/replaceable on site: we try to avoid using very specialized manufacturing techniques and choose the simpler solutions, which we can replicate in the competition area in reasonable time
- as search and rescue robots are considered “expendable” in a disaster scenario, a cheaper solution for the same problem would give the ability to use more robots at the same time. So we are trying to use cheap off-the-shelf parts and avoiding the dependency on expensive sensors.
- we are developing the hardware in an agile environment (adopted from software development), so we do not spend months on cad and planning. Instead we iterate every week and we are planning and building proof-of-concept robots in a continuous pipeline, in order to validate the ideas. Then we apply the lessons in the next design phase and we test the next generation accordingly. This method allows us to always have working prototypes and parts thus not risking the not-being-ready-to-move situation a waterfall based project often results in.

A. Hardware

The primary goal is to never get stuck. The all wheel drive construction with no other surface to touch the ground combined with the ability to change the wheelbase and move the center of mass gives the robot extreme maneuverability (Fig. 5). The second task is to learn how to efficiently drive/control this configuration to solve the tasks.

A *static wheel hub* design (Fig. 6) allows us to use larger wheels with double bearing and also to protect sensitive parts inside the rotating surface. With the ability of quick wheel replacement, the vehicle can always be adapted to specific terrains and challenges in minutes (Fig. 7).

For *motors and gears* we were using cheap off-the-shelf servos in the early prototypes last year, but after a few iterations we gave up as we calculated the speed and torque requirements to drive our large and exotic wheels in every situations. In 2015 we found the Banebots 256:1 planetary gears and their motors to match our needs perfectly and they worked so well, we will use them in 2016 too.

Fig. 5. The planned 2016 robot has a twisting body with an arm attached on one end

Fig. 6. Static wheel hub design

Fig. 7. Rolling through the stepfield

For high performance *on-board computing* we have decided to use an Odroid XU4 single board computer as it gives excellent computing power compared to its size. This versatile 8 core ARM-Linux computer runs a fully fledged ROS and has abilities to control the robot alone on autonomous missions.

Fig. 8. RoboCup German Open 2015 Best in class manipulation 2nd place

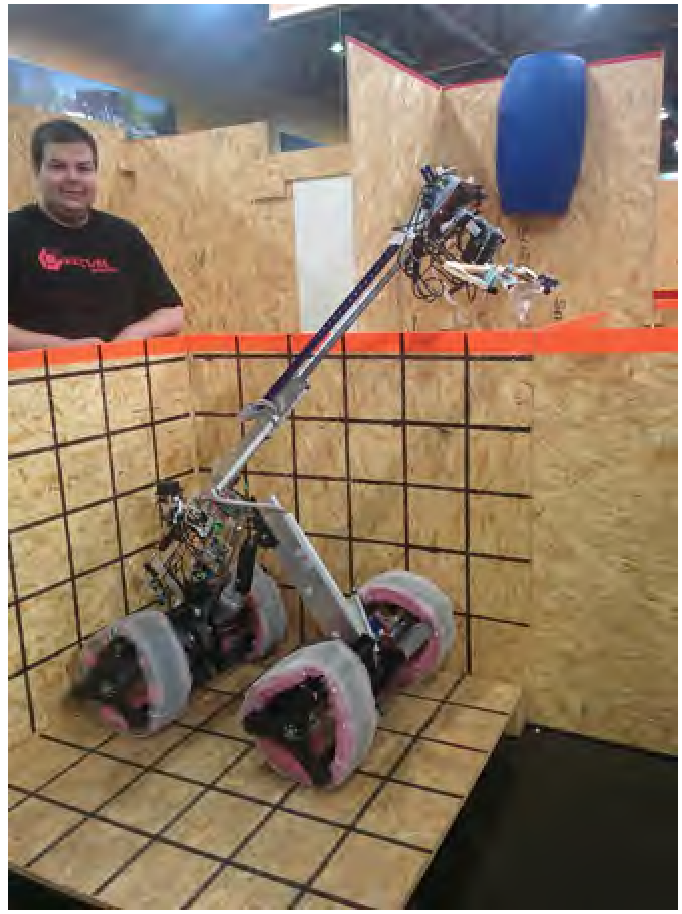

Fig. 9. The variable geometry with a linear neck mechanism

Fig. 10. The arm has linear actuators, with grippers attached

In the case of resource exhaustion we are planning on decentralizing the robots processing to a number of computers, distributing the tasks and IO devices to dedicated hardware.

For *low-level operations* and where real-time operation is a requirement we use TI's Tiva-C ARM-Cortex M4 microcontroller. This excellent MCU has a great variety of hardware peripherals and is responsible for collecting data from various low-level sensors and also to command the drive motors. The communication between the Tiva and the Odroid is handled by the Rosserial ROS-package.

We have decided to use Lithium-Polymer high discharge *batteries* for their superb capacity/price ratio and as we have never had any issues with them. The total on-board capacity is 10 Ah providing the robot over 30 minutes of continuous operation in normal scenarios. We use several high-efficiency voltage converters to supply the adequate voltages/currents for each subsystem.

The 2015 "giraffe" robot (Fig. 9) had a freedom of 5 degrees to move the sensor head into correct position and orientation to examine the possible victims. Based on last year's experience, we are trying to give the robot lower profile while moving, so we opted for a base robot with a robotic arm (Fig. 10).

The *robotic arm* works like a crane with a rotating base and an extendable telescopic arm that has a lightweight pan-tilt sensor head and also actuated grippers to enable the robot to open doors, turn valves and solve various pick and place tasks (Fig. 10). Our replaceable head actuators enable us to optimize the robot for specific tasks in short time.

For *cameras in visible light range* we plan to use Sonys PS-Eye cameras which have a well deserved fame among developers for their excellent frame-rates and low-light sensitivity. Our choice for localization/navigation was an RP-Lidar, a simple yet effective laser scanner providing 5 Hz measurements in 5 meter range.

For outdoor missions our laser scanner data is fused together with the measurements of a u-blox M8T multi-GNSS receiver providing 10 Hz absolute positions. To achieve centimeter-level accuracy we plan to establish a short-baseline RTK system by operating a reference station in situ.

A structure.io *depth camera* gives us 3D perception abilities and it provides the information needed for detecting 3D structures, and movement. We will continue to use Flir's Lepton thermal camera sensor modules for their outstanding capabilities.

For *inertial measurements* and pose sensing the Invensense's MPU-6050 6-DOF IMU coupled with a Honeywell HMC-5983 compass are still top-notch devices in their categories

Fig. 11. Sensory head with thermal camera, RGB camera, depth camera and ultrasonic, CO_2, inertial sensors

Fig. 12. Operator station RoboCup German Open 2015

despite their age.

Of course the robot also has some extra sensors/devices like microphone arrays, speakers, a CO_2 and an air pressure sensor, and numerous robot state encoders/potentiometers built in. We primarily use the Tiva-C microcontroller boards to communicate with these low-level digital (over I2C and SPI) or analog devices, or USB for the higher level protocols.

Please see Tables **??** as well as Table III in the Appendix.

B. Software

The general software framework in use is the open source Robot Operating System, ROS, version Indigo[1].

For simulation our intention is to use Gazebo 4 simulation platform enabling us to quickly and easily test new design concepts and software features[2].

[1] http://www.ros.org/
[2] http://gazebosim.org/

For mapping we have decided to use the Hector SLAM module of ROS developed by Stefan Kohlbrecher and Johannes Meyer at TU Darmstadt[3].

For GNSS positioning we have good experiences with Tomoji Takasus RTKLIB, an open source GNSS positioning software[4].

For pose calculations using inertial sensors we use Sebastian Madgwicks open source implementation of Robert Mahonys DCM filter[5]. We have plans for developing an own-rolled EKF solution for sensor fusion using Matlab.

OpenCV is an excellent and proven tool for image processing and feature detection[6].

The PCL (Point Cloud Library) contains routines for the manipulation of sets of discrete points, and allows the user to detect surfaces and predefined objects well[7].

Please see Table IV in the Appendix.

C. Communication

For communication between our hardware nodes not connected physically we plan to rely solely on TCP/IP communication over wireless 802.11ac compliant network devices running on 5 GHz channels. These equipments provide a great combination of reliability, bandwidth and cost. Being provided a private channel is a key for reliable communication in an environment of a robotics contest with noise levels making any radio communication barely eventual. When placing ROS nodes on computers our goal is to keep networking load at the minimum, so every computer should process the information of their connected sensors and just send the deducted information to the other interested parties. We have also developed a solution of a low overhead mission logging concept that collects the data required for re-enacting the missions while keeping network load minimum.

D. Human-Robot Interface

The human-robot interface is still under heavy development, there is not much we can disclose at the moment. However we have set the following design guidelines:

- the operator system/interface should be as simple as it could be (this also helps quick and easy deployment in rural/disaster areas)
- the operator system should actively and autonomously analyze and evaluate the incoming information thus off-loading the operator by filtering information relevant to the actual situation

III. Application

This section covers the practical aspects of our system.

[3] http://wiki.ros.org/hector_slam
[4] http://www.rtklib.com
[5] http://www.x-io.co.uk/open-source-imu-and-ahrs-algorithms
[6] http://opencv.org/
[7] http://pointclouds.org

A. Set-up and Break-Down

All our robots and operator station was designed with quick and easy deployment capabilities in mind. The typical set-up time for robots is under five minutes, the time requirement for the operator station is around ten minutes, so our complete system can achieve operational state in about 10 minutes (assuming the processes executed in parallel). We also have made efforts to let the robot and operator station work uninterrupted (even when replacing batteries), so no cold restart is required between missions at all.

The break-down process consists of steps with more or less the same time requirements. The overall break-down process should take no more than 5 minutes under normal circumstances.

B. Mission Strategy

Our intention is to develop a versatile robot being capable to compete successfully in all Robocup Rescue standard scenarios, but with the prioritized order of: mobility-manipulation-autonomy. We believe that autonomy is a great feature to decrease the mental load of the operator (and to enable a single person to operate multiple robots). However, at disaster recovery the human decision making could save lives, so we strive to create a hybrid semi-autonomous solution that allows human and computer minds collaborate in a well-balanced and effective way.

C. Experiments

For evaluating our robots' performance we have tried to replicate some of the key elements of the common mobility and perception tasks based on the NIST standard test methods[8].

D. Application in the Field

While working on the development of RoboCup Rescue robots, our team members continuously learn new skills and apply those either in their academic, industrial or personal life. We are dedicated to share these experiences with the following young generations of engineers, so we have an educational/motivational mission, to give them reason to learn programming, to solder leds to a microcontroller, or to learn the math required for every applied scientific career path.

IV. Conclusion

As a new RoboCup Team we do have a lot to learn, and we are still working hard to create a community in Hungary where robotics interested people can gather together. We believe that our efforts could bring industry, academia and elementary/high schools together, and create an accelerated path where applied science students could gain invaluable experience every day.

[8]http://www.nist.gov/el/isd/test-methods.cfm

Appendix A
Team members and Their Contributions

- Zoltán Abonyi embedded computing, robot operator
- Mátyás Borvendég electrical engineering, manufacturing
- Dávid Dudás mechanical engineering, 3D printing
- Péter Gliga electrical engineering, manufacturing
- Gábor Guta computer vision, decision making
- László Jaskó mechanical engineering
- Péter Kopiás team lead, robot concept, software development
- Márton Krauter localization, inertial measurements, public relations
- Péter Maricza mechanical engineering, modelling
- Miklós Márton software development, embedded computing
- Zoltán-Csaba Márton 3D perception, object recognition
- Imre Petrovszki electrical engineering, software development
- István Szikra systems integration, software development

Appendix B
CAD Drawings

Please see Fig. 13 and 14, along with the earlier drawings.

Fig. 13. Rendered view of our CAD drawing

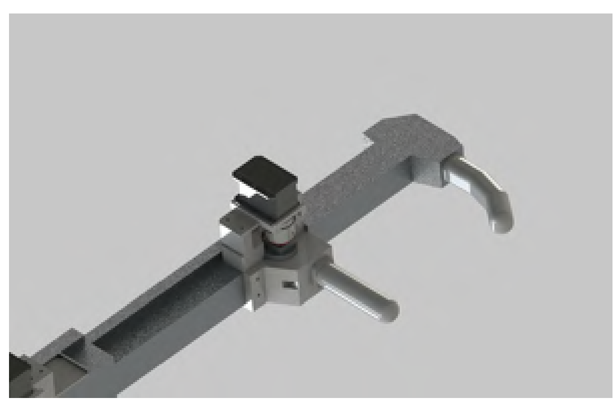

Fig. 14. Rendered close-up view of our arm

APPENDIX C
LISTS

A. Systems List

TABLE I
MANIPULATION SYSTEM

Attribute	Value
Name	Barrel16
Locomotion	4 WD
System Weight	20 kgs
Transportation size	100 x 100 x 50 cm
Typical operation size	100 x 100 x 50 cm
Unpack and assembly time	10 min
Startup time (off to full operation)	5 min
Power consumption (idle/ typical/ max)	50 / 240 / 2400 W
Battery endurance (idle/ normal/ heavy load)	4 / 1 / 0,5 h
Maximum speed (flat/ outdoor/ rubble pile)	5 / 5 / 3 km/h
Payload (typical, maximum)	10 kg max
Arm: maximum operation height	160 cm
Arm: payload at full extend	300 grams
Support: set of bat. chargers total weight	1 kg
Support: set of bat. chargers power	160W
Support: Charge time batteries (80% / 100%)	1 / 1.2h
Support: Additional set of batteries weight	1 kg
Any other interesting attribute	Dynamic wheelbase
Cost	3000 EUR

TABLE II
OPERATOR STATION

Attribute	Value
Name	Operator station
System Weight	20 kg
Weight including transportation case	30 kg
Transportation size	60 x 60 x 60 cm
Typical operation size	100 x 50 x 50 cm
Unpack and assembly time	10 min
Startup time (off to full operation)	5 min
Power consumption (idle/ typical/ max)	200W
Battery endurance (idle/ normal/ heavy load)	120 / 90 / 60 min
Any other interesting attribute	Puppet Manipulator
Cost	1500 EUR

B. Hardware Components List

TABLE III
HARDWARE COMPONENTS LIST

Part	Brand & Model	Unit Price	Num.
Drive motors	Banebots RS540	8 EUR	4
Drive gears	Banebots P60	60 EUR	4
Motor drivers	Non-brand	30 EUR	4
Smart servos	Turnigy S518D	40 EUR	6
DC/DC	Non-brand	30 EUR	2
Batteries	Turnigy 5000mAh	50 EUR	2
Battery chargers	Turnigy 80W	50 EUR	2
Microcontroller unit	TI Tiva C	20 EUR	2
Computing unit	Hardkernel Odroid XU4	100 EUR	2
Wifi adapter	TBD		2
IMU	Drotek 6-DOF IMU	15 EUR	2
Compass	Drotek HMC-5983	10 EUR	2
GNSS	Drotek u-blox M8T XXL	90 EUR	2
USB camera	Sony PS Eye	30 EUR	4
Infrared camera	Flir Lepton	200 EUR	2
Lidar	RPLidar	400 EUR	1
CO2 sensor	Non-brand	20 EUR	1
Operator laptop	Lenovo T430	1000 EUR	1

C. Software List

TABLE IV
SOFTWARE LIST

Name	Version	License	Usage
Ubuntu	14.04	open	OS
ROS	Indigo	BSD	Framework
OpenCV [1]	2.4	BSD	Computer Vision, image-based recognition
PCL [2]	1.7	BSD	Scene segmentation and object recognition
Hector SLAM [3]	0.3.4	BSD	Localization and Mapping
RTKLIB	2.4.3	BSD	Localization

ACKNOWLEDGMENT

We would like to express our gratitude for their contributions towards the following individuals and organizations:

- Datastart Ltd.
- Former members: Bence Magyar, Árpád Pálóczy, Sam Pfeiffer
- ABC Center for Intelligent Robotics: dr. László Kutor, dr. Zoltán Vámossy, dr. András Molnár
- University of Óbuda: dr. Lászl Kutor, Gusztáv V. Tényi, Tamás Sándor, dr. György Schuster
- John von Neumann Computer Society: dr. Zoltán Istenes
- Attila Sipos: http://magyarokamarson.hu
- Technische Universität Darmstadt: Prof. Dr. Oskar von Stryk, Stefan Kohlbrecher
- Robocup Rescue League Organizing Team
- BHG Jégpálya
- Weisz Müanyagipari Kft.
- Robotshop.com
- Function demoscene meeting http://function.hu
- QbParty demoscene meeting: http://qbparty.hu

REFERENCES

[1] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[2] 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.

[3] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.