Preprint of the paper which appeared in:
IEEE Intl. Symposium on Safety, Security, and Rescue Robotics (SSRR),
21-26 Oct. 2013, Linkoping, Sweden.

Overview of Team ViGIR's Approach to the Virtual Robotics Challenge

Stefan Kohlbrecher[1], David C. Conner[2], Alberto Romay[1], Felipe Bacim[3], Doug A. Bowman[3], and Oskar von Stryk[1]

www.teamvigir.org [1] *Simulation, Systems Optimization and Robotics Group, Technische Universität Darmstadt, Germany*
[2] *TORC Robotics, United States* [3] *Center for Human-Computer Interaction, Virginia Tech, United States*

***Abstract*—With the DARPA Robotics Challenge (DRC), a call to an ambitious multi-part competition was sent out to the robotics community. In this paper, we briefly summarize the approach for addressing the Virtual Robotics Challenge (VRC) where software for control and supervision of a capable humanoid robot must be developed. Team ViGIR, comprising members from the US and Germany, leveraged previous robotics competition experience and a variety of open source tools, to achieve sixth place in the VRC out of 126 registrants, thereby advancing to the next round of the DRC and obtaining an Atlas robot.**

I. Introduction

The Fukushima Daichi nuclear plant disaster once again showed that disaster response and mitigation could improve significantly if more capable robots were available for use. This motivated the DARPA Robotics Challenge[1]. In the VRC a simulated humanoid robot has to perform a series of tasks inspired by real world needs. Details of the challenge and participation models can be found online. Seven top scoring teams in the VRC received an Atlas robot being developed by Boston Dynamics and proceed to the next round of competion. For developing and integrating the complete onboard and offboard software within the few months available until the VRC, we decided to reutilize and adapt available, preferably open source, software from within and outside the team.

II. System Overview

Team ViGIRs software uses ROS as middleware [1]. While the DRC simulator[2] was locked to use the ROS fuerte version during competition, ROS groovy was used for all team software to leverage recent software advancements only available for ROS groovy, for example the MoveIt! motion planning framework [2]. With developers spread over two continents, development was tracked using the Redmine project management system, frequent conference calls, and an on-site sprint meeting preceding the VRC competition.

III. Basic Capabilities

For the VRC, the team focused on developing the basic capabilities required for operation of a humanoid robot in the competition scenarios. Due to space constraints, other software capabilities like *footstep planning*, *behavior control*, *semi-autonomous grasping* and *open-loop motion editing* are not detailed in this paper.

[1]http://www.theroboticschallenge.org/
[2]http://www.gazebosim.org/wiki/DRC

Operator Control Station (OCS) The OCS is a set of graphical user interface components that allow the operator to control the robot and visualize information about its current state and the world surrounding it. The goal for OCS design was a customizable user interface with support for different widgets that can be changed and reorganized based on the task at hand. Three main components to the OCS were used in each task: a 3D perspective view widget that allows the operator to control end effectors directly, visualize 2D and 3D reconstructions of the environment, and plan robot motion; a top-down orthographic view widget that is used for navigation and to request more information about the environment; and a camera widget that allows the operator to request images with a requested resolution from every camera in the robot. The other OCS components available to the operator include grasp control, individual joint control, and planner configuration tools. The complete setup used in the VRC is shown in Fig. 1.

Fig. 1: OCS setup used in the VRC

Bandwidth-Constrained Communication Only very limited communication is allowed between the robot and the Operator Control Station (OCS). In the VRC, traffic was restricted by emulation software and teams could only transmit a limited amount of data to the robot (upload) and from the robot (download). The most strict scenario allowed only 7Mb of data download from the robot over the whole 30 minute mission time, making bandwidth management and careful selection of data to be transmitted a serious issue. The overhead of communication when using a single ROS Master to connect the onboard and OCS systems was prohibitive; therefore, a communications bridge using a compressed transport based on Google Protobufs and bz2 compression was developed. Bandwidth constraints were never exceeded during VRC missions.

Perception Due to bandwidth limitations, only limited amounts of perception data could be transmitted to OCS, so fusion of data and conversion to a low-bandwidth representation were very important. LIDAR point cloud data is fused using PCL tools [3] and the probabilistic octomap approach

[4], reducing data volume compared to conventional pointcloud storage. Octomap data can then be sliced in a region of interest over a given height range, resulting in a 2D occupancy grid map representation. Using this approach, map data for obtaining general situational awareness can be transmitted to the OCS with minimal bandwidth cost.

Manipulation of the environment was required for 2 of the 3 VRC tasks and thus a key capability. Limited bandwidth means that the transmission of a full 3D model of the environment is prohibitive, so collision free motion planning has to happen onboard the robot. To this end, the operator can specify a intended joint or cartesian endeffector target pose for the robot and motion planning including full 3D obstacle avoidance is performed using the MoveIt! motion planning framework [2] onboard the robot, leveraging full 3D obstacle information. Grasps for different kinds of object are template based and are generated offline using the GraspIt! [5] toolkit.

Motion Control A per joint PID controller using feedforward motion compensation was employed for simplicity and robustness. It can be used to follow joint trajectories and will also stitch trajectories for smooth transition between them. Keyframe based motions worked very well in simulation. Basic locomotion controllers were provided for optional use to teams.

IV. Task Performance

Approaches to the three VRC tasks and resulting performance are summarized. For each task, 5 instances with varying conditions were used and for each of those, up to 4 points could be scored depending on the degree of task completion.

Task 1 (Driving) After walking up to the vehicle using the capabilities presented in Section III, the driving task presented two challenges. Getting into the car was achieved by using a carefully designed motion that makes use of mechanical effects to reduce uncertainty in positioning the robot into a sitting posture in the car (Fig. 2a). Actuation of the car controls was difficult, as the seat was modeled as a sticky surface. When performing the required manipulation in contact with the environment, the robot would slip in the seat unpredictably. This resulted in loss of sitting posture in all runs.

Task 2 (Rough Terrain) Due to a bug in the optionally provided walk controller which only showed up during VRC, it was not possible to get into walking mode again when the robot has stood up after a fall. Therefore, after a fall during a task in the VRC, crawling was the only option left for locomotion. Open-loop key frame-based quadrupedal locomotion had been developed which was used in the VRC during the rough terrain task and whenever the robot had fallen. After walking up to the first gate, the robot encountered a simulated mud pit, which could not be traversed using the provided walking controller. Therefore, the robot was commanded into a sitting posture and backwards crawling was employed as the mode of locomotion for the remainder of Task 2 (Fig. 2b).

Task 3 (Hose Manipulation) The most challenging part of the hose manipulation task was aligning the hose with the standpipe after picking it up (Fig. 2c). While picking the hose up worked reliably using our approach, moving it towards the location of the standpipe was more challenging, as movement of the hose was unpredictable when dropped on the table for re-grasping with the other arm. For this reason, strategy was switched to walking with hose in hand during VRC, which allowed alignment with the standpipe in one run.

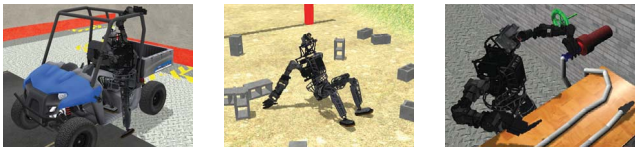

Fig. 2: VRC Tasks: a) Driving (left), b) Rough Terrain (middle) and c) Hose Manipulation (right)

V. Evaluation

From out of 126 Track B and C teams registered for the VRC, 26 passed qualification and 22 scored in the VRC. With a total of 27 points (Table I) Team ViGIR was ranked 6th behind 5th and 4th with 29 and 30 points and before 7th and 8th with 25 and 24 points. The distribution of points scored in the three tasks was quite similar for teams ranked 3rd to 8th.

	Run 1	Run 2	Run 3	Run 4	Run 5	Total (Max. 20)
Task 1	0	1	1	1	0	3
Task 2	4	2	4	4	4	18
Task 3	1	1	2	1	1	6

TABLE I: Team ViGIR VRC scores.

VI. Conclusion

In this paper, a short overview of Team ViGIRs entry to the Virtual Robotics Challenge is presented. The team is currently transfering and adapting the software developed for the VRC to the actual Atlas robot and extending it. It plans to provide as much as possible of their software developments as open source to foster progress in rescue robotics.

Acknowledgment

The authors would like to thank all team members who contributed and continue to contribute to the success of the team: Tony Angell, Florian Berz, Lindsay Blassic, Thorsten Graber, Jesse G. Hurdus, Alex Little, Johannes Meyer, Jochen Mück, Karen Petersen, Philipp Schillinger, Dorian Scholz, Jacob Sheppard, Alexander Stumpf and Ben Waxler.

References

[1] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.

[2] S. Chitta, I. Sucan, and S. Cousins, "Moveit!" *IEEE Robotics Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[3] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (pcl)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–4.

[4] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[5] A. T. Miller and P. K. Allen, "GraspIt! a versatile simulator for robotic grasping," *Robotics & Automation Magazine, IEEE*, vol. 11, no. 4, pp. 110–122, 2004.