Teleoperation System Design of Valve Turning Motions in Degraded Communication Conditions

Stéphane CARON, Yoshihiko NAKAMURA
Department of Mechano-Informatics, University of Tokyo, Japan

1. Introduction

During the DARPA Robotics Challenge[1] (DRC), robots were expected to solve a number of tasks under teleoperation by a human operator. Limits in execution time and teleoperation bandwidth required teams to implement some level of autonomy on their robots, yet meaningful input could still be provided by the operator on a regular basis via the team's Operator Control System (OCS). An OCS is an execution-level interactive human-robot teleoperation system, often integrated in a single Graphical User Interface (GUI) to reduce the number of switches for the operator.

The purpose of the present paper is to report on the development of Team Hydra's OCS for the DRC in the context of the valve-turning task. We describe the design of the system as well as the technical choices made, meanwhile pointing out the underlying research questions and directions for future work.

For the robot middleware, we chose ROS[2] (the "Robot Operating System"), on top of which we developped all of our system components. The GUI in particular was integrated to RViz, the visualization software from ROS, which comes with various extension mechanisms such as panels and interactive markers. Panels are areas of the GUI where one can add arbitrary widgets connected to arbitrary commands. Interactive markers will be detailed later on.

To realize the valve-turning task, we have developed a number of components, split between two categories corresponding to perception and execution. Perception components include the construction of an environment model, identification of the valve position, and estimation of contact locations. Execution components include walking pattern generation, inverse kinematics and trajectory generation.

2. Perception Components Design

2·1 Environment Perception

We build a 3D point-cloud model of the environment using a 2D scanning-type laser range sensor UTM-X002S (Hokuyo Automatic Co. Ltd) tilted along the pitch axis by a MX-64 motor (Dynamixel). For this purpose, we used the ROS packages `hokuyo_tilter` and `pointcloud_tools` implemented by Kiyoshi Irie (Chiba Institute of Technology). Calibration between the environment model with respect to the humanoid's kinematic chain was done by manual identification of the kinematic transform between the sensor base frame and the robot's body. This approach has limited precision, as validation is done by a human operator, and is sensible to joint-calibration errors. Consequently, we prepared for execution-time updates: when limbs of the humanoid appear in its field-of-vision, the operator can tune the transform so as to match them with the kinematic model.

Here, the human operator implements a feedback loop between the point cloud and the robot's kinematic model. This feedback loop could be automated, *e.g.*, by sampling points on the robot's 3D mesh and using point registration methods to match them in the point-cloud. The underlying problem is to calibrate the sensor's base transforms with respect to the robot's kinematic chain. Recent works such as [3] have demonstrated the feasibility of an autonomous solution to this problem using visual markers for simultaneous calibration of several sensory inputs.

2·2 Valve Identification

The position, orientation and size of the valve are input by the human operator with feedback from the point cloud. To enable this input, we developed a model-fitting interface based using a custom RViz panel and the `interactive_markers` library[3]. In this library, a "marker" consists in:

- a 3D mesh or a primitive shape (*e.g.*, a box),
- a set of "controls",
- a right-click menu.

The marker controls are graphical handles by which the operator can translate or rotate the mesh along any of its degrees of freedom (see Figure 1).

Algorithm 1 Model-fitting Procedure

1. Add the valve marker
   (button from the valve panel)
2. Enable marker controls
   (right-click menu of the valve marker)
3. Translate and rotate the valve mesh to match the position and orientation given by the point cloud
   (marker controls)
4. Adjust the scale of the mesh
   (slider from the valve panel)
5. Iterate 3 and 4 until fitting is complete
6. Disable marker controls
   (right-click menu of the valve marker)

[1] `http://www.theroboticschallenge.org/`
[2] `http://www.ros.org`
[3] `http://wiki.ros.org/interactive_markers`

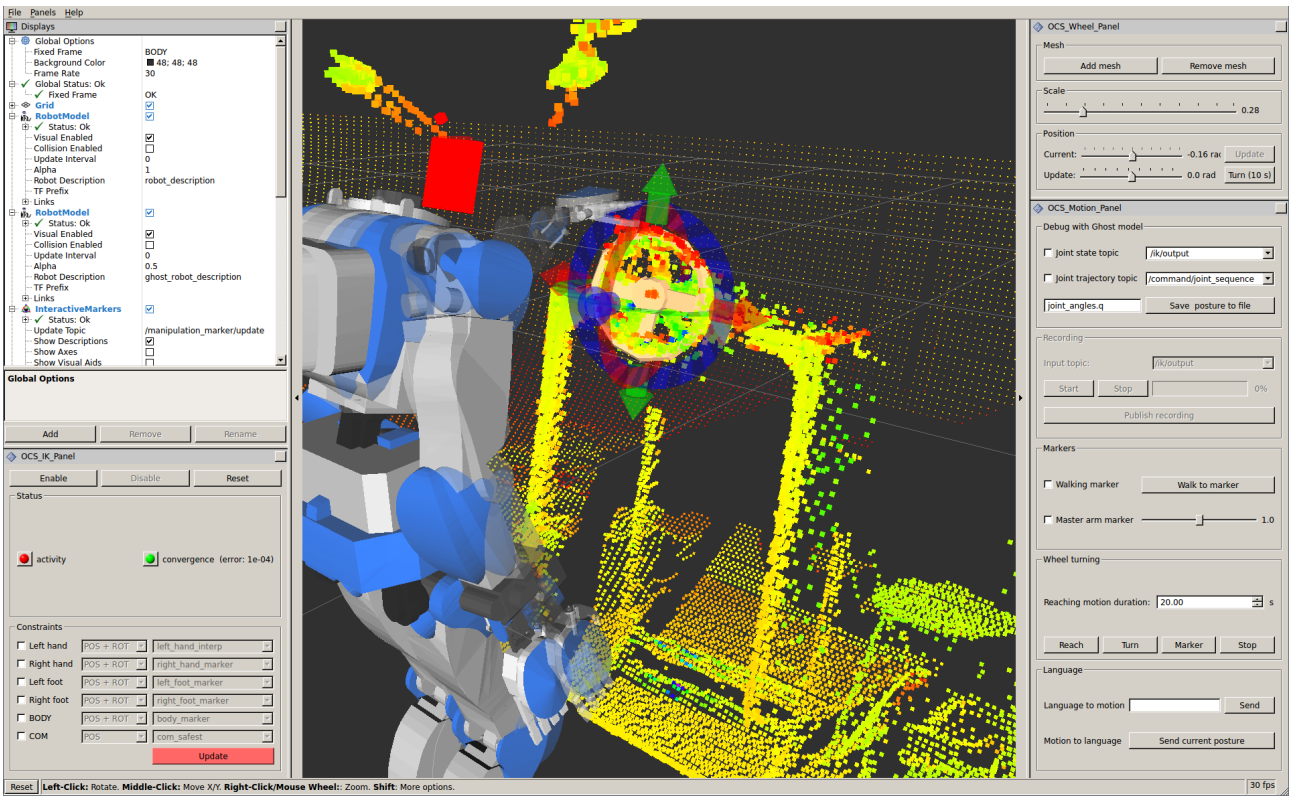

Fig.1 Interface of the Operator Control System with a model of the HRP4-R humanoid robot. Its head has been replaced with an adjustable plate (in red) for the laser-range sensor frame. Panels on the left: default "Displays" from RViz and a custom panel for IK control. Panels on the right: Ghost model, Walking, Language and Wheel. The ghost model is used to preview IK results before execution on the real robot. Walking markers can be used to control the Walking Pattern Generator [5]. The language panel is used to query a large human-behavior database for postures [6]. Finally, the wheel panel allows to add/remove the valve marker, update its scale, send the reaching and turning commands. It displays two sliders: one for the estimate of the current wheel rotation, and the second for the desired turn angle. (Points above the robot head are artifacts from the sensor.)

The operator then follows the model-fitting procedure described in Algorithm 1. Overall, there are seven parameters to fit: the mesh's scale, position and orientation.

Although the complete execution of the procedure takes less than a minute to a trained operation (see the accompanying video [2]), this task ought to be solved by the software as well, *e.g.*, using point-cloud registration methods [4].

2·3 Contact Location

The next step is to estimate the contact locations on the valve. To enable natural input of contact coordinates with respect to a 3D mesh, we developed a wrapper to `interactive_markers`, the `manipulation_markers` library, which is publicly available online.[4] Manipulation markers consist of

- a 3D-meshed "parent" marker, and
- a set of primitive-meshed "contact" markers.

Contact markers are used to calculate the end-effector poses (*i.e.*, position and orientation) by which the robot makes contact with their parent mesh. They can be added on-the-fly by the operator via the right-click menu. Because their kinematic transform are defined in the parent marker's reference frame, moving the parent marker (in our case, turning the valve) automatically updates the position and orientation of each of its contact markers.

3. Execution Components Design

Our turning strategy relies on a single contact between the palm of the robot's left hand and the outer rim of the valve. To avoid the need to move the feet while turning, we further decided to apply only quarter-turns at a time, with the robot periodically reaching for the top of the valve to start the motion again.

3·1 Walking Pattern Generator

First of all, the robot has to walk to the valve. To achieve this purpose, we used the Walking Pattern Generator (WPG) from [5], which allowed us to abstract the task to the 2D ground position of the Center-of-Mass (COM). Using an interactive marker

[4] `https://github.com/Tastalian/manipulation_markers`

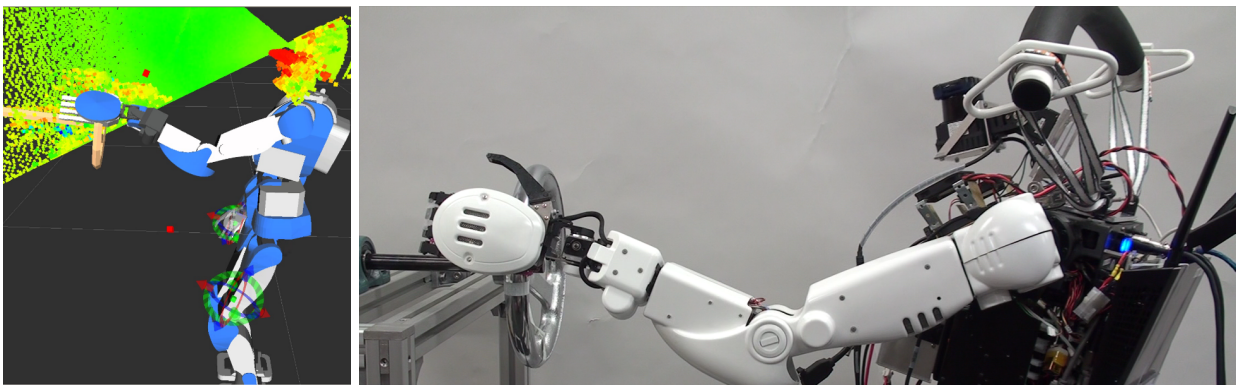

Fig.2 Simultaneous view of the experiment in the OCS (left) and from a video camera (right). The Hokuyo laser-range sensor is mounted on top of a fixed plate set on top of the humanoid's chest. Complete execution of the task took around 10 minutes to the human operator, including on-the-spot recovery of a loss of contact after three quarter-turns. See the accompanying video [2].

on the foot horizontal place, the operator evaluates the point-cloud data before inputting the goal location of the COM to the WPG. Then, after a 10-second refresh interval, he or she can reiterate the process until the robot is standing in front of the valve. Once a satisfactory location is reached, we fix the foot locations on the ground and turn the valve with no additional call to the WPG.

3·2 Inverse Kinematics

The kinematic transform of a contact marker with respect to the robot's body frame can be used to establish contact via Inverse Kinematics (IK) and position control. For the former, we used RoKi [1], a software library from the Motor Intelligence Lab. (Osaka Univ.). Given a set of end-effector poses (*i.e.*, position and orientation) and a COM position (maintained above the foot support area for stability), it could perform the whole-body inverse kinematics of our 34-DOF robot model within milliseconds.

3·3 Interpolation of Reaching Trajectory

Using the reduction from joint-space to workspace coordinates provided by the IK, we only control the left hand pose for the rest of the task. When the operator sets the contact marker on the valve, we compute two via-points from the current hand pose to the contact pose. The first one lies between the hand and the valve. The second one is closer to contact and has a z-coordinate *above* that of the contact point in order for the left hand to make contact with a downward-pointing incidence vector. Finally, a 20-second trajectory is interpolated between the four left-hand poses (initial, via-point 1, via-point 2, contact) by using linear interpolation for the end-effector position and spherical linear interpolation (Slerp) for its orientation.

3·4 Control of the Valve Turning Motion

Once contact is made, we generate the valve turning motion by turning the valve manipulation marker in the GUI and forwarding the resulting IK output (where the left hand is bound to the contact marker) to the position controller. However, to cope with the limited bandwidth constraint, we further cut the operator input to a single turning angle. As depicted in Figure 1, to execute of a quarter-turn, the operator sets the desired angle ("Update" slider) and presses the command button. The relative angle is then the only data sent to the robot field PC over the network, at which point the latter will update its internal model of the valve and perform the IK locally.

4. Valve Turning Experiment

We tested our framework with an HRP4-R humanoid robot and a steel valve mounted on a metal frame. Figure 2 shows a side-by-side comparison of the simultaneous OCS view and real state of the robot during operation. In the laboratory environment, it took the operator 10 minutes to execute the task. See the accompanying video [2].

Position estimation errors accumulate during execution of the task. In the present experiment, after reaching a third time for the top of the valve, the humanoid did not properly establish contact. The operator could notice that the valve did not turn with the hand (*e.g.*, by looking at its rims in the point cloud view) and adapt to the situation by translating or downscaling the wheel mesh in the OCS.

Contact state estimation ought to be done at the robot level. Torque-controlled humanoids, or position-controlled ones with force-torque sensors in their wrist, can estimate the contact force directly. The estimation is still possible with previous generation humanoids like HRP4 where force-torque sensors are typically located in the ankles. Yet, it is more involved as forces are only observed after propagation along the kinematic chain.

5. Conclusion

In this paper, we described the various components implemented by Team Hydra for the valve-turning task of the Darpa Robotics Challgenge, from perception to motion generation. We detailed how we adapted our design to cope with the limited teleopration bandwidth available during the challenge. Finally, we evaluated our OCS through experiments on the HRP4 humanoid platform.

One of the goals of robotics research is to make robots more autonomous. As such, some operator tasks that we described in this paper ought to be done autonomously by the software. These tasks include the fitting of the valve to the point cloud, computation of the via points for the reach task, detection of contact and potential slippage, etc.

Acknowledgments: This work was supported by the New Energy and Industrial Technology Development Organization (NEDO), The International R&D and Demonstration Project on Robotic Field / Research and Development of Disaster-Response Robot Open Platform (FY2014–FY2015).

The source code for the valve-turning OCS includes ROS packages developed by Kiyoshi Irie (Chiba Institute of Technology) for the Hokuyo sensor and Dynamixel motor, and by Yasuhiro Ishiguro for the integration between ROS and OpenRTM-aist.

References

[1] RoKi - Robot Kinetics library, Motor Intelligence Lab, Osaka University. Available online at `http://www.mi.ams.eng.osaka-u.ac.jp/software/roki.html`, July 2015.

[2] Accompanying video. Online at `https://scaron.info/research/rsj-2015.html`, July 2015.

[3] Oliver Birbach, Udo Frese, and Berthold Bäuml. Rapid calibration of a multi-sensorial humanoids upper body: An automatic and self-contained approach. *The International Journal of Robotics Research*, 34(4-5):420–436, 2015.

[4] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009.

[5] Carlos Santacruz and Yoshihiko Nakamura. Analytical real-time pattern generation for trajectory modification and footstep replanning of humanoid robots. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2095–2100. IEEE, 2012.

[6] Wataru Takano and Yoshihiko Nakamura. Integrating whole body motion primitives and natural language for humanoid robots. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 708–713. IEEE, 2008.