J Intell Robot Syst
DOI 10.1007/s10846-015-0256-5

From Autonomy to Cooperative Traded Control of Humanoid Manipulation Tasks with Unreliable Communication

Applications to the Valve-turning Task of the DARPA Robotics Challenge and Lessons Learned

Calder Phillips-Grafflin · Halit Bener Suay · Jim Mainprice · Nicholas Alunni · Daniel Lofaro · Dmitry Berenson · Sonia Chernova · Robert W. Lindeman · Paul Oh

Received: 21 October 2014 / Accepted: 28 July 2015

Abstract In this paper, we present our system design, operational procedure, testing process, field results, and lessons learned for the valve-turning task of the DARPA Robotics Challenge (DRC). We present a software framework for cooperative traded control that enables a team of operators to control a remote humanoid robot over an unreliable communication link. Our system, composed of software modules running on-board the robot and on a remote workstation, allows the operators to specify the manipulation task in a straightforward manner. In addition, we have defined an operational procedure for the operators to manage the teleoperation task, designed to improve situation awareness and expedite task completion. Our testing process, consisting of hands-on intensive testing, remote testing, and remote practice runs, demonstrates that our framework is able to perform reliably and is resilient to unreliable network conditions. We analyze our approach, field tests, and experience at the DRC Trials and discuss lessons learned which may be useful for others when designing similar systems.

Keywords Humanoid robotics · Manipulation · Teleoperation

C. Phillips-Grafflin (✉) · H. B. Suay · J. Mainprice · N. Alunni · D. Berenson · S. Chernova · R. W. Lindeman
Worcester Polytechnic Institute, Worcester, MA, USA
e-mail: cnphillipsgraffl@wpi.edu

H. B. Suay
e-mail: benersuay@wpi.edu

J. Mainprice
e-mail: jmainprice@wpi.edu

N. Alunni
e-mail: nalunni@wpi.edu

D. Berenson
e-mail: dberenson@wpi.edu

S. Chernova
e-mail: soniac@wpi.edu

R. W. Lindeman
e-mail: gogo@wpi.edu

D. Lofaro · P. Oh
Drexel University, Philadelphia, USA
e-mail: dan@danlofaro.com

P. Oh
e-mail: paul@coe.drexel.edu

1 Introduction

The 2011 disaster at the Fukushima Daiichi nuclear power plant illustrated the limitations of then state-of-the-art disaster response robotics. This led to the 2013 DARPA Robotics Challenge (DRC) Trials, where roboticists were invited to compete on eight tasks representative of those encountered in a disaster recovery scenario. Each of these eight tasks was defined to require a highly mobile and dexterous robot. Humanoid robots, while not required for the trials, are especially well suited for these

Published online: 03 September 2015

tasks as they take place in human environments. To match conditions experienced in real-world disaster situations, trial rules specified that communication between robot and operators would be restricted in bandwidth and suffer from high latency. This prevents operators from working with sensor feedback at a high rate, limiting situation awareness, and making it difficult to accurately monitor the result of the robot's actions in real time.

This paper does not focus on providing novel fundamental methods for robot control. Instead, we focus on the development and testing of a framework for the seventh DRC task – turning industrial valves (shown in Fig. 1). Our DRC team was structured so that separate groups of people worked on each DRC task, which allowed development to proceed in parallel (see [12, 20, 28, 32, 37, 53, 54] for a description of our DRC team's work on other tasks). While the work described in this paper targets the valve-turning task, many of the components of our framework, including both software components and operating procedures, were adopted by other member of our team for other DRC tasks.[1] The valve-turning task poses a number of particularly difficult challenges, which we analyze in Section 3. However, the core challenge is manipulating an object whose general shape, but not size and location, is known *a priori*. Thus, we need a system which is able to navigate the robot to a location where it can manipulate the object of interest, determine grasp locations, and dynamically generate trajectories for manipulation.

Our system is composed of five primary components: (1) an operator-guided perception interface which provides task-level commands to the robot, (2) a motion planning algorithm that autonomously generates robot motions that obey relevant constraints, (3) an "unreliable communication teleoperation" toolkit, which we have released as an open-source ROS package [33], that limits the traffic on the data link and makes the system resilient to network dropouts and delays, (4) an operational protocol that dictates how the team of operators must act to operate the robot, and (5) a testing process that simultaneously tests the system and serves to train the human operators.

[1] Several software components, especially the robot–workstation datalink, were used directly by other parts of our DRC team, while other components, such as the user interface and operational protocol, were modified to match their tasks.

Using our system, operators can specify the manipulation task in a straightforward manner by setting the pose and dimensions of the object to be manipulated in a 3D display of the pointclouds acquired by the robot, shown in Fig. 2. The system then lets operators plan a feasible trajectory, and once validated using a previewing system (also shown in Fig. 2), send it to the robot for execution.

This paper describes our system design, performance, and lessons learned. In particular, we discuss the development of our system design from an initial focus on autonomy to the cooperative traded control system used in the DRC Trials. We discuss lessons learned through the design process that could be useful to others, including why some standard techniques could not surpass the ability of well-trained operators on critical tasks, such as localizing objects in pointcloud data or monitoring errors in trajectory execution. We also discuss the performance and limitations of our motion planning and control approaches as well as alternative approaches for performing the task.

2 Related Work

Prior work in the area of disaster recovery robotics [4, 9, 47, 50] has revealed the need for better group organization, perceptual and assistive interfaces, and formal models of the state of the robot, state of the world, and information as to what has been observed. Here, we focus on the problem of teleoperating a single humanoid robot remotely under unreliable communication.

2.1 Teleoperation of Humanoids

While a fully autonomous disaster-recovery robot would be desirable, no fully-autonomous humanoid robots currently exist. Instead, human supervision (i.e. teleoperation) is necessary to perform complex tasks in unstructured environments. Teleoperating a humanoid robot involves controlling actions such as head motions, grasping and manipulation of objects, navigation, speech, gestures, etc. The case of teleoperating a humanoid robot is particularly difficult due to the high number of degrees of freedom (DoFs) and the constraint to maintain balance. Teleoperation for humanoid robots is a rather new area of research. A

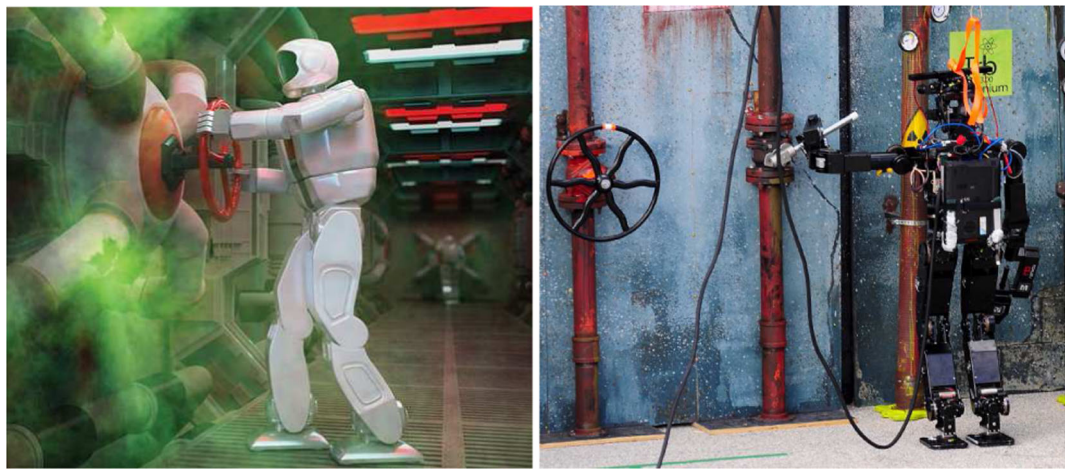

Fig. 1 Hubo2+ (*left*) in a simulated environment and DRCHubo (*right*) at DRC Trials turning a valve

recent survey on the topic can be found in [18], in which authors list three types of challenges to teleoperate humanoids, two of which are relevant for disaster recovery: 1) challenges created by physical properties of humanoids (e.g., high DoFs, morphology) and 2) operator-based challenges (e.g., situation awareness, skill, and training). In this paper, we present a framework that addresses both types of challenges.

There are a number of architectures that attempt to tackle the problem of mobile manipulation [10, 23, 44]. Mobile manipulation tasks, be they performed by a humanoid or wheeled robot such as the PR2, consist of several difficult problems: where to place the robot in relation to the object to be manipulated [15, 46, 51], how to grasp the object [5, 25, 30], and how to plan the robot's movements [6, 22, 26]. Despite extensive work in this area, to the best of our knowledge, there is no available framework for humanoids robots tailored for operator-guided object manipulation in unstructured environments with limited communication to the robot.

2.2 Managing Autonomy in Teleoperation

All teleoperated robots require a certain level of autonomy. Managing this autonomy is crucial in

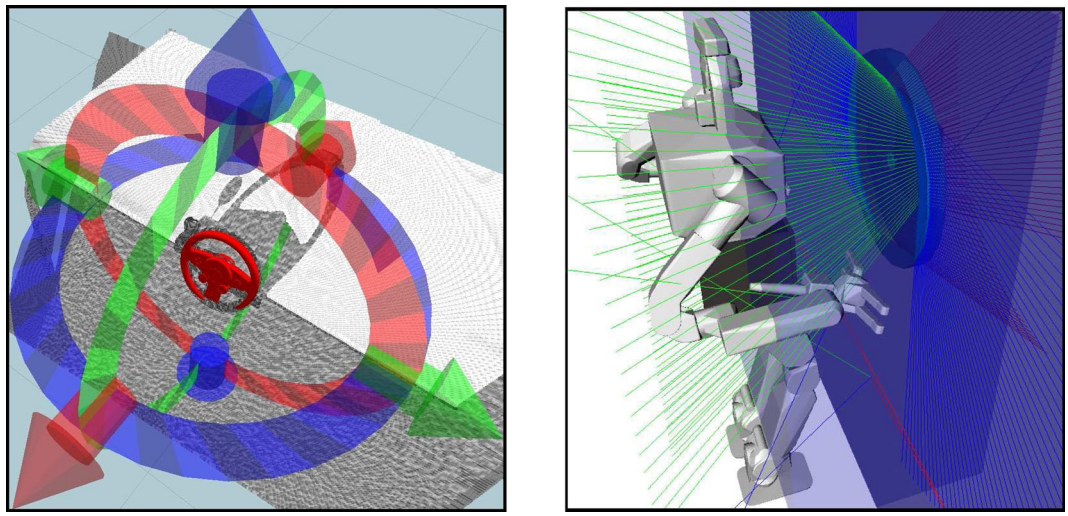

Fig. 2 The operator identifies and localizes a valve in a pointcloud using an interactive marker (*left*), visualizes the motion planned to maximize the turn angle of the valve by testing multiple hand placements before execution (*right*)

the design of a teleoperation system. Supervisory control [17], defined by Sheridan [43] as a process in which "one or more human operators are intermittently programming and continually receiving information from a [robot] that itself closes an autonomous control loop", provides a good framework to classify different control approaches. However, other terminologies and methodologies have since emerged to describe ways of managing the robot's autonomy, the three most relevant for our task are defined in [18]:

Direct control
: The operator manually controls the robot, minimal autonomy is involved on the robot side. The robot is controlled in a master-slave interaction. An example of a direct-control approach is the control of each DoF of a manipulator using a joystick.

Traded control
: In traded control the operator and the robot both control the robot's actions. The operator initiates a task or behavior for the robot. The robot then performs the task autonomously by following the desired input while the operator monitors the robot. For instance, in [39], an interactive robot is teleoperated by selecting predefined tasks for the robot to perform. Our approach is based on traded control as the operator only specifies the target valve and its properties to initiate the task, which is then performed by the robot using autonomous motion planning and execution.

Collaborative control
: This mode corresponds to high-level supervision where the robot is considered to be a peer of the operator. The role of the operator shifts from an operator who dictates every movement, to a supervisor who guides at a high-level. This approach is often used for unmanned systems controlled from a central command post [3, 29].

When a team of operators controls a robot in any of those modes the strategy is called "Cooperative". The method presented in this paper is a form of *Cooperative Traded Control*. While a comparative study between different teloperation methods is not within the scope of this paper, in Section 6 we discuss the evolution of our approach and compare it to alternative control strategies.

2.3 Teleoperation with Low Bandwidth

Highly unstructured disaster environments, such as those we target in this work, provide a challenge where communication can be difficult due to the unknown properties of the building materials, making transmission and reception of signals unreliable [31]. While communication over unreliable channels has been studied extensively for many years, in particular Shannon's seminal work [42], we are concerned with controlling a robot over a limited-bandwidth network where the underlying unreliability of the channel has been mitigated by the use of TCP. Low-bandwidth communication covers a broad range of research, which can be categorized by the amount of delay that the system attempts to handle. Typically, these categories are roughly 0–2 seconds, 2–10 seconds, and greater than 10 seconds latency [8]. For instance, many surgical systems operate between zero and two seconds of latency sometimes across distant locations [27]. Latency greater than two seconds is typically found in research related to earth orbit or farther systems, such as Lunar robots [8, 34]. Latency greater than ten seconds extends even farther including the Mars rovers, which have a delay of many minutes [7]. The system we present in this paper is meant to operate seamlessly with a latency between zero and five seconds, with packet loss and periodic dropouts that are analogous to communication in a demolished building.

2.4 Service-Oriented Architectures

The system we present has been developed within a Service-Oriented Architecture (SOA). An SOA is a system architecture that consists of discrete software modules that communicate with each other. SOAs have become a popular choice for robotics since they allow the software to be highly modular and adaptive [35]. A range of SOAs are currently available, including Microsoft Robotics Developer Studio (MRDS) [24], Joint Architecture for Unmanned Systems (JAUS), Hierarchical Attentive Multiple Models for Execution and Recognition (HAMMER) [40], and Robot Operating System (ROS) [36]. We chose ROS for our system due to its extensive proven ability to control high-DOF robots such as the PR2. ROS has also been applied to more anthropomorphic humanoid

robots such as the Nao [1], Robonaut 2 [16], and TU/e TUlip [21]. Additionally, ROS was chosen for its extensive libraries, such as TF, which maintains the transformations between all frames of the robot, and its built-in visualization tool (RVIZ) which allows for fast user interface development. However, ROS does not perform well in unreliable network conditions. We describe how we overcame this in Section 4.2.

3 Problem Description

The valve-turning task of the DRC poses a number of significant challenges to a robot. In the task itself, the robot must perceive the location, size, and pose of the valve, compute a suitable placement, and turn the valve. Besides these characteristics of the task, the time-limited and competitive nature of the DRC Trials imposes additional challenges of communications, supervisory control, and testing.

3.1 Perception

The robot must be able to reliably locate the target valves in a potentially-unstructured environment. In the worst case, the robot must be able to locate valves of unknown size and shape, in the presence of unknown obstacles, with a range of ambient lighting and visibility. This requirement is extremely challenging, particularly since the perception subsystem must be robust enough to be suitable for competition use.

3.2 Base Placement

In the presence of obstacles, finding a robot placement suitable for completing a manipulation task (such as turning the valve) requires finding a) a base/foot placement and b) a set of configurations for turning the valve that allow the robot to maintain balance. Base/foot placement defines the shape and the location of the support polygon that the robot needs to stay in balance throughout the task. This is a problem intertwined with the problem of finding a set of configurations for the manipulation task, since the projection of the center of mass of the robot must lie in the support polygon at all times for balance.

3.3 Manipulation

The core challenge of the valve-turning task is manipulation of the valve. In our approach, manipulation in this task consists of two stages; first, the motion generation stage, which must generate trajectories that obey constraints of balance and closed-chain kinematics, and second, the execution of these trajectories.

For legged and high-DoF robots such as DRCHubo, movements of the upper body can make the robot unbalanced. As a result, it is important to consider the center of mass when generating the robot manipulation motions. However, to be able to turn valves with high friction and stiction, the robot must use its maximal capabilities (i.e., using both arms). Thus, the hands must move in a coordinated manner. Our motion planning component, based on the CiBRRT algorithm [6], is able to account for those constraints given the pose and shape of the manipulated object.

3.4 Operation

A critical and wide-reaching design choice for the system is the selection of the operator control approach. This choice involves not just a selection of operating mode from those discussed in Section 2.2, but the development of the operator interface and operational protocol. An important consideration present in the DRC is that the operator interface must be robust enough for competition use and efficient enough to complete a time-limited task.

3.5 Communication

As specified in the DRC rules, communications between the robot and supervisor workstation are both limited in bandwidth and subject to high latency. This poses a considerable challenge to development and operation. Ideally, sensor data from the robot, consisting primarily of joint values, camera images, and pointclouds, would be promptly transmitted to the user, yet the ability to send this data is extremely limited by poor network conditions. Given the time limits, simply waiting for data to transmit is not a viable solution. To account for this, the system must be designed to minimize bandwidth use, and the system components (including the role(s) of human operators) must be devised to require as little data as infrequently as

possible. Specifically, for the DRC, DARPA specified bandwidth between 1 Mbit/sec and 100 Kbit/sec, with latency between 100 milliseconds and 1000 milliseconds, intended to simulate the unreliable and varying communications available in a disaster zone [48].

3.6 Testing

Since the DRC Trials are a competition, not only must the entire system be tested, but operators must also be trained to effectively and efficiently operate the robot. Given the limited development time available, this system testing/operator training must take place concurrently with system development. In light of system complexity and shifting challenge rules, devising a test procedure that is representative of challenge conditions is difficult. The training process must produce operators who are experienced with not only robot behavior, but robot behavior *in error conditions*, something that often requires real-world tests to uncover.

An important component of devising such a testing procedure is the method used to assess performance; this must cover not just the individual software and hardware components on the robot, but the overall performance of the teleoperated robot system, including both hardware, software, and the human operators. In the context of the DRC, we assessed the performance of our system using scoring criteria inspired by the DRC rules (before the publication of the final criteria) and specified by DARPA (after the publication of the final scoring criteria) [48], as discussed in Section 5. This scoring criteria specifies both tasks to complete (i.e. turning each of a set of valves) and time limits on setting up for and completing the tasks.

4 Framework Description

Our manipulation framework is intended for high-DoF robots. We have applied a preliminary version of it to PR2 and Hubo2+ robots, as shown in Fig. 3 [2]. In this paper, we focus on the final version applied to the DRCHubo robot, developed by the Korean Advanced Institute of Science and Technology (KAIST). DRCHubo, an evolution of the Hubo2+ humanoid, has two 6 DoF legs, two 7 DoF arms, and a 2 DoF head. The hands each possess three fingers, with all fingers controlled by a single DoF. DRCHubo is equipped with a sensor head containing a tilting Hokuyo LIDAR for providing pointcloud data and three cameras for stereo vision, configured to provide three different stereo baselines as needed.

4.1 Architecture Overview

The system architecture, shown in Fig. 4, shows the data flow through the system. Software modules running on the robot are shown on the left (yellow), while those running on the operator workstations are shown on the right (blue).

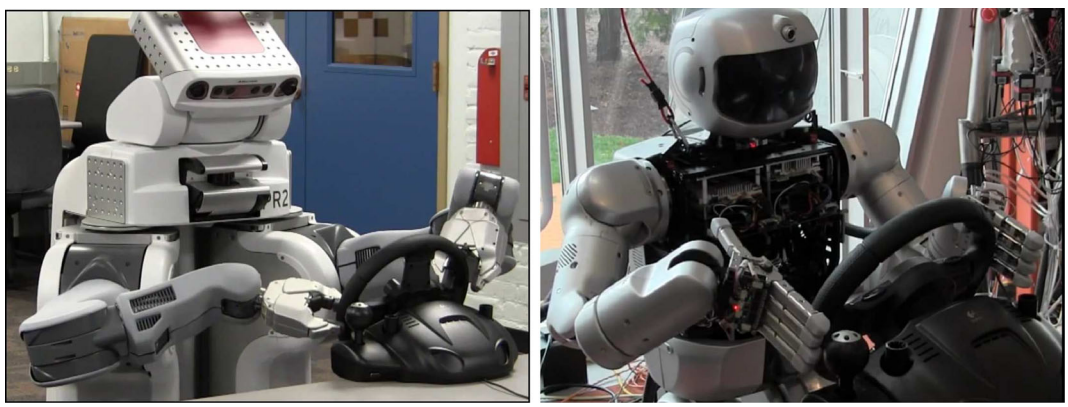

Fig. 3 Preliminary versions of our framework were applied to the PR2 robot (**a**) and the Hubo2+ humanoid (**b**)

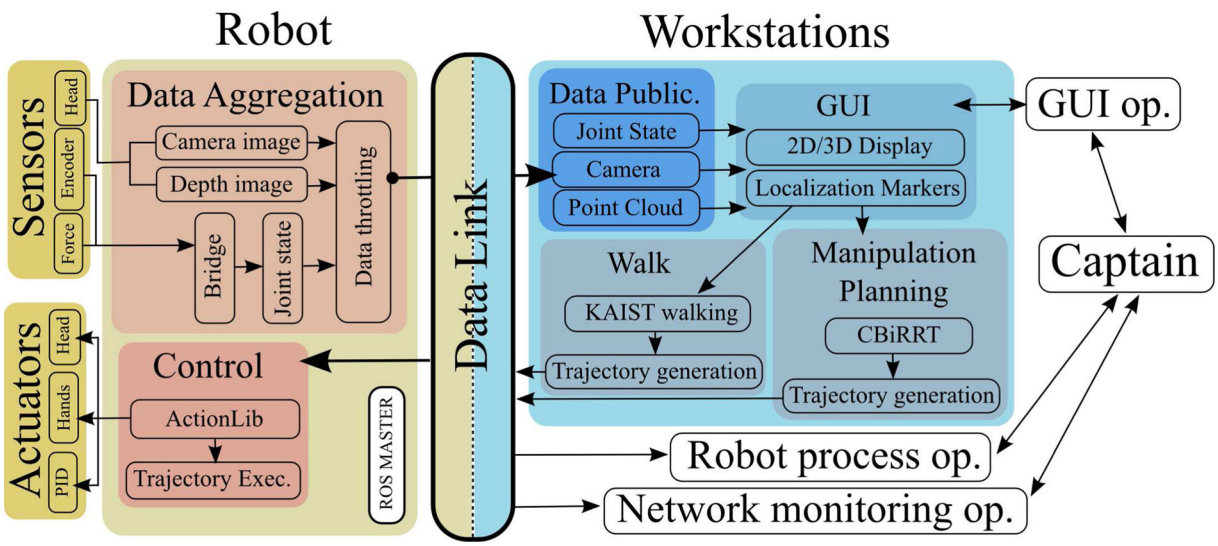

Fig. 4 System diagram showing data flow through the framework. Operator interaction in *white boxes*

On the robot, the data aggregation module reads sensor output and packages it into compact messages. The control module receives trajectories and then monitors their execution.

On the workstations, the Graphical User Interface (GUI) displays data received from the data aggregation system to the operators. The GUI allows an operator to specify the object pose and dimension and send commands to the motion generation systems. Finally, the motion trajectories from the walking generation and the manipulation planning systems are sent to the robot through the data link.

The implementation of the framework relies on ROS [36] for the communication between modules, RViz for the user interface, and OpenRAVE [14] for the motion planning and pre-visualization of trajectories. The walking generation code is based on KAIST's Rainbow walking framework. All data transmission between the robot and the workstation happens over ROS, however we have implemented our own data-link software to throttle data rates and limit communication to only necessary information.

The primary function of the data-aggregation system is to reformat sensor data used on the workstation so that communication across the restricted datalink is limited. As shown in Fig. 4, it simultaneously processes camera images, pointclouds, encoder values, and force sensors allowing the framework to be highly modular and quickly adaptable to different robots, as well as being suitable for both real and simulated environments. The system can also be reconfigured during operation to handle changes in the available sensor data, such as selecting an alternate image or pointcloud source, or changing the quality of data received.

For communication with Hubo's motor controllers, we use Hubo-Ach, a real-time control daemon that uses a high-speed, low-latency IPC called Ach [13]. Hubo-Ach implements a real-time loop in which all of the motor references and state data are set and updated respectively. The bridge component of the data aggregation system combines joint angle and motor control sensor values read through Hubo-Ach and republishes them as ROS messages. The head sensors (i.e., camera and LIDAR for DRCHubo), are controlled through ROS nodes.

4.2 Robot–Workstation Data Link

The data link between the robot and operator workstation is responsible for the transfer of sensor data from the robot to the operator(s) and commands from the the operator(s) to the robot. To simplify development, as with robot and workstation, this data link uses ROS.

This means the software running on robot and workstation behave as a single distributed system. ROS is a combination of a distributed node-based inter-process communication (IPC) system and a family of libraries built atop it. Natively, ROS IPC is a single-master system in which a single master node coordinates inter-node communications. In distributed ROS systems, this means that one of the computers (usually one aboard or connected to the robot itself – in our use, the robot) runs the master node. Experimental multi-master systems exist in which multiple master nodes are responsible for separate sets of nodes. Of particular importance among the libraries included in ROS is TF, which provides transformations between all frames of a robot via the *TF tree*.

Standard ROS systems, be they single-master or multi-master, are vulnerable to network problems. Low-bandwidth and high-latency conditions like those experienced in the DRC can (and, in our experience, do) result in the failure of time-sensitive operations such as TF queries and the loss of synchronization in synchronized data. The latter is a problem particular to ROS; camera images and the relevant camera model information are transmitted separately but rely on synchronization via timestamps. Synchronization failure for these messages results in the failure of all operations attempting to use the camera data.

Equally troublesome, ROS provides limited built-in functionality for data throttling and de-duplication. Natively, each node subscribed to a particular data topic receives its own copy of the data. In distributed systems, this means that duplicate copies of the same data will be be sent over bandwidth-constrained network links. Similarly, there is no way for nodes to directly control the rate at which they receive data – for example, in our system, joint state data is produced at the same rate as the real-time loop aboard the robot (200 Hz), a far higher rate than that needed by the motion planner on the control workstation.

While we considered both single- and multi-master ROS architectures, we selected a single-master design due to the development simplicity and familiarity it offered. This choice came at the cost of significant development to address the limitations of ROS and single-master systems. Our solution to these limitations, consisting of a dedicated toolkit for degraded networks [33], allows for robust single-master ROS systems over network conditions like those experienced in the DRC (and worse) without imposing additional constraints or limitations on developers. In particular, our toolkit addresses several critical issues:

TF
: The high bandwidth demands and sensitivity to latency preclude the use of a single TF tree for the robot and workstation. Instead, we use separate "divorced" trees; one tree is generated directly on the robot, while the second is generated on the workstation from periodic joint state updates. This approach greatly reduces the bandwidth demands of TF, as static transforms are *never* transmitted, and joint state updates are considerably smaller (and sent far less frequently) than the equivalent transforms.

Reliable transport and throttling
: Data transport between robot and workstation is provided by rate-limited relays that replace the standard publisher-subscriber model. These relays, based on ROS's non-persistent service calls, replace the simple equivalents provided in ROS and implement automatic detection of network failures, notification and warnings to the operator, and automatic recovery of broken network sockets. Relays provide fine-grained rate control, ranging from request-only to free-flowing – however, to avoid flooding the network, they only transmit new data after previous transmissions have completed. While considerably different in implementation from standard ROS topics, these relays expose the same basic interfaces and require no modifications to existing ROS nodes. To improve performance, generic relays are used for lightweight data such as joint states, while image and pointcloud-specific relays are used as necessary.

Synchronization
: Synchronization of paired topics is solved by our rate-limiting relays. These relays ensure that paired messages are synchronized prior to transmission, and by transmitting them together, they ensure that synchronization is maintained until delivery.

Data compression
: Reduction in bandwidth needed for image data is achieved with a combination of resizing and JPEG compression implemented in the image relay. This combination results in a factor of 1000 reduction in size for images (from approximately 5 MB uncompressed to 5 KB compressed). In the pointcloud relay, pointclouds produced from LIDAR scans are

Table 1 Data sizes for different compression ratio and frequency of transmission

Data	No compression	ratio 1	ratio 2	Freq. used
Joint State	0.82 KB	–	–	10Hz
Camera	5120 KB	230 KB	4.97 KB	0.5Hz
Pointcloud	10 MB	≈ 100 KB	≈ 50 KB	On demand

compressed using a voxel filter and a selection of pointcloud compression algorithms[2] including ZLIB and PCL's Octree-based compression [38]. Using this approach, we can reduce the data usage of pointclouds by over 95 %. For both images and pointclouds, the compression process is completely transparent to other nodes and requires no modifications to existing code.

In addition to the development of a dedicated toolkit to improve performance on degraded networks, we limited the bandwidth demands of our system by aggressively limiting the rate of data transfer. From our remote testing experience, we were able to reduce the frequency of data transfer to the minimum required for task completion. Table 1 reports the data sizes and frequency used for communication between the robot and workstation. The compressed camera frame correspond to lower resolution (320x240) with JPEG compression quality for ratio 1 and ratio 2 of 50 and 20, respectively. The compression of pointclouds corresponds to the compression algorithms mentioned above with no voxel filtering for ratio 1 and filtering with voxel size of 0.02 m for ratio 2.

4.3 User Interface

When teleoperating the robot, the operator must be able to monitor the robot state as well as its surroundings to maintain situation awareness. Thus, our GUI, shown in Fig. 5, provides monitoring capabilities through 2D camera images as well as a 3D display of the robot configuration and pointcloud data. The user can switch what data is displayed on screen using RViz's built-in features. In addition to sensor data, the GUI displays the motion planner error conditions and the control system's state through panels, text and color codes.

The operator controls the robot by specifying a set of parameters that are sent to the motion planner and the walking generation module (i.e., end-effector pose, turn angle, etc). We use interactive markers [19] and control panels to determine and input those parameters. Distance and direction for the walking generation are determined using an interactive marker. Valve size, turn amount, and choice of manipulator for the task are determined using both an interactive marker and the control panels. Before querying the motion planner all parameters can be verified at a glance by looking at the control panels, which greatly reduces the possibility to incorrect commands being sent to the robot.

Interactive markers (see Fig. 2) provide six-DoF handles, three translation DoFs and three rotation DoFs, which enable the operator to quickly define a pose in a 3D display. Since we use interactive markers to simultaneously select and localize the object to manipulate, we avoid the use of complex object detection and localization algorithms, instead relying on the operator for these capabilities. The shape attached to the interactive marker can be a box, a disk, or a triangle mesh. To specify the pose and dimensions of an object (e.g., lever or disk valve) the operator aligns the shape to pointcloud data using the interactive marker. In the DRC, when localizing the valve for walking to a standing position in front of it, the pose estimate does not have to be as precise as for manipulation, usually only requiring mild accuracy in terms of distance from the robot to estimate the walking distance. Hence the average time to align the marker over one trial while operating the robot are 9.3 secs for walking and 41.6 secs for manipulation (when a much more accurate alignment is needed).

Once the object is localized and the planner parameters are selected, the operator can send a planning request. The resulting motion can be pre-visualized at will in a dedicated 3D GUI component (see Fig. 2). This phase limits potential mistakes made by the operator as well as dangerous robot behavior.

In addition to operator input and feedback, the GUI controls the data flow over the unreliable link to the robot. The operator can request pointclouds and turn

[2]Full details of the pointcloud compression algorithms available, including a comparison of performance and features, can be found in the documentation of [33]

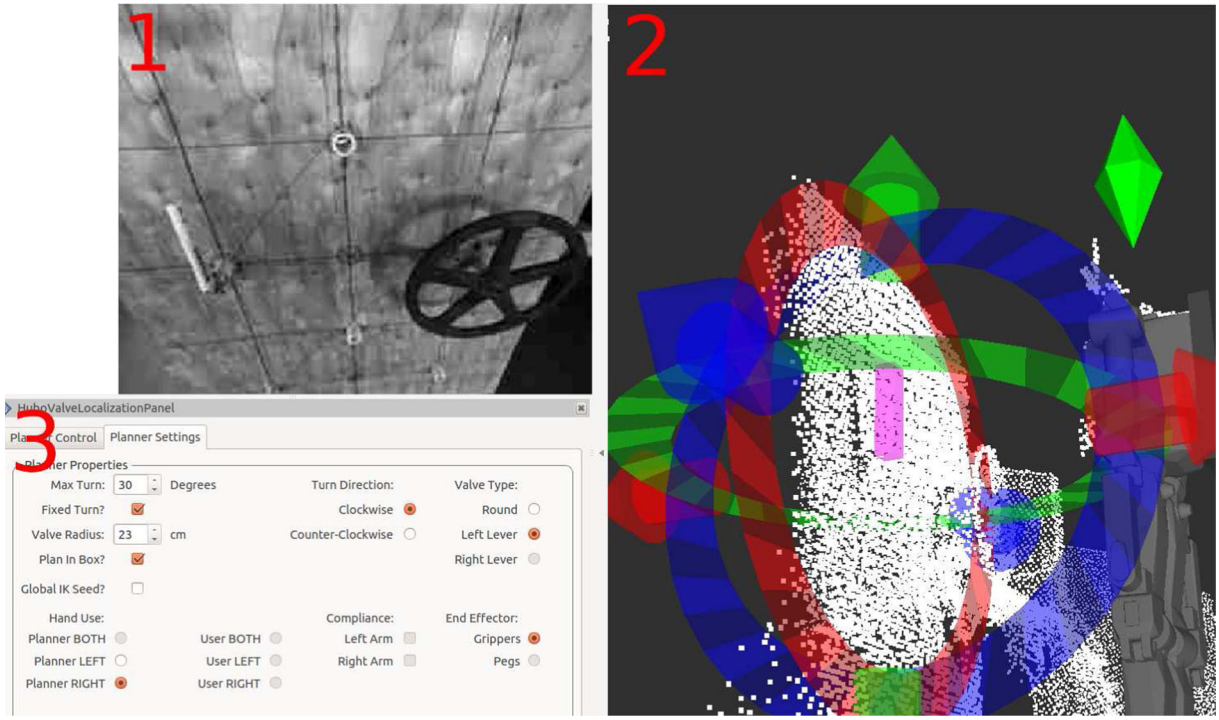

Fig. 5 Screen capture of the Graphical User Interface (GUI). **1** video feed of the lever and round valve, **2** display of pointcloud and interactive marker, **3** planner-settings panel

on and off the camera image request loop. Thus data from the robot is transmitted only when necessary to minimize communication.

4.4 Motion Planning and Execution of Trajectories

Once the object pose and dimensions are set by the operator, the operator can generate the robot's motion using the motion planning component. The paths produced by the motion planner are collision-free and respect end-effector pose and balance constraints. After validation by the operator, the trajectories produced by the planner are sent through the data link and executed by the control system aboard the robot.

4.4.1 Motion Planning

For valve turning, each manipulation task involves three subtasks : 1) **Ready**: a fullbody motion that sets the robot's hands close to the valve ready to grasp it, with knees bent, lowering the center of gravity to be more stable 2) **Turn**: An arm(s) motion that grasps the valve, performs a turn motion, releases the valve and returns to the initial configuration (so it can be repeated without re-planning, assuming that the environment is static) 3) **End**: fullbody motion that brings the robot back to the walking configuration. While these motions are specialized for valve turning the motion planner can be easily reconfigured to manipulate other objects by inputting a different set of constraints.

The motion planning component of the system is built upon the CBiRRT algorithm [6], which is capable of generating constrained quasi-static motion for high-DoF robots with balance constraints. While a number of motion planning algorithms are capable of planning constrained motion [45, 49], we chose CBiRRT for its explicit incorporation of balance and closed kinematic chain constraints in addition to support for end-effector constraints. All three types of constraints are essential to the valve turning problem – without any one of them, the robot would fall over, fail to turn the valve, or damage itself. CBiRRT generates collision-free paths by growing Rapidly-exploring Random Trees (RRTs) in the configuration space of the robot while constraining configurations to configuration-space manifolds implicit in the constraints. Average planning

Table 2 CBiRRT average and stdev planning time in seconds for DRCHubo on the three valve turning subtasks

Valve type	Ready	Turn	End
Lever	1.91 (1.11)	0.99 (0.39)	1.72 (0.91)
Circular	2.71 (1.12)	2.72 (1.55)	2.38 (1.72)

time of CBiRRT for each subtask are reported in Table 2.

In the valve turning task, the motion must obey constraints defined by the valve pose provided by the operator (as explained in Section 4.3). The end-effector pose constraints are specified as Task Space Regions (TSR) [6]. A TSR consists of three parts:

- T_w^0 : transform from the origin to the TSR frame w;
- T_e^w : end-effector offset in the coordinates of w;
- B^w : 6×2 matrix of bounds in the coordinates of w:

In our implementation for valve-turning we have defined three tasks that the robot can perform: 1) Turn a lever with the right hand 2) Turn a lever with the left hand 3) Turn a circular valve with both hands. Each of these tasks corresponds to a TSR constraint definition.

In all cases, iterative Jacobian pseudo-inverse inverse kinematics are performed to find a whole body configuration given the location and radius of the manipulated object. The TSRs are then defined according to the hand locations when grasping the object and the pose of the object. For instance, the TSR for one-arm lever motions is defined as follows:

$$T_w^0 = T_w^{valve}$$

where T_w^{valve} is the valve pose in the world.

$$T_e^w = (T_w^{valve})^{-1} * T_w^H$$

where T_w^H is the hand pose in the world when grasping the valve.

$$B^w = \begin{bmatrix} 0 & 0 & 0 & \theta & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

where θ is the desired rotation angle of the lever. When planning for full-body motions, we also define TSRs for the feet to keep their position and orientation fixed in the world. In order to perform large turns on the circular valve, we have implemented an algorithm that iterates through interpolated hand placements along the valve to find valid start and goal IK solutions that maximize the turn angle (see Fig. 2).

4.4.2 Trajectory Execution and Control

The path generated by the motion planner is first retimed using piece-wise linear interpolation before being sent over the data link to the control system. Trajectories are executed aboard the robot by feeding waypoints at 200Hz to the on-board controllers, which track the waypoints with PID controllers running at 1KHz. The operator is informed of the end of the trajectory execution by a monitoring system based on a ROS Action Server that returns success or failure if the robot has reached the end of the trajectory in the time constraints.

4.5 Human-Robot Interaction and Team Command

Due to the number of modules, the complexity of the system can easily generate too much cognitive load on a single operator. Single-operator use of our system is possible – indeed, many of the remote tests were done primarily by a single operator – but tasking a sole operator with simultaneously monitoring and controlling the robot was inefficient and led to mistakes. To reduce errors and improve the efficiency of operation, we defined a multi-operator scheme that distributes different parts of the task among multiple operators. In our multi-operator approach, each member is assigned a particular function (see Fig. 6), and we make use of checklists and a "playbook", summarizing failure cases and possible strategic decisions to be made, to dictate the operators' tasks. To clarify responsibility for decisions and improve responsiveness in failure cases, we adopted an explicit chain of command and responsibility between the various operators.

The team roles were the following:

Captain
: Dispatches the different sub-tasks to the other operators and keeps track of the current strategy (e.g., the order in which to perform the environment scans, where to walk, and the manipulation tasks). Effectively, the core function of the Captain is to maintain task-wide situation awareness and convert this knowledge into timely commands. This operator should have a good understanding of the system as well as precise knowledge of the primary and alternate strategies developed *a priori* in the "playbook."

GUI operator

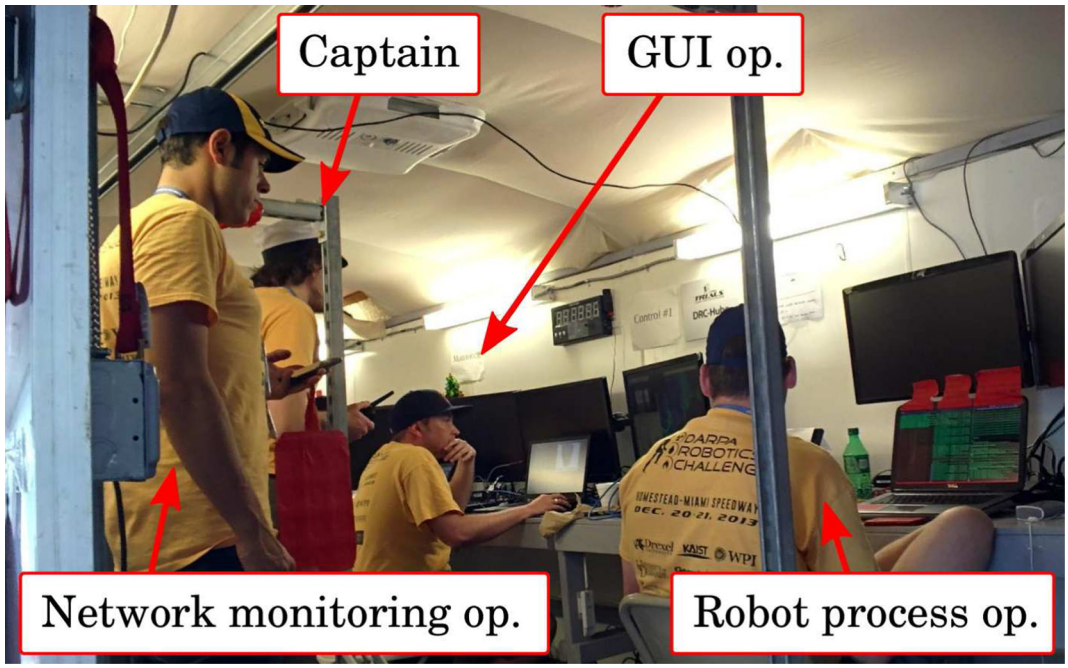

Fig. 6 Operator roles while teleoperating DRCHubo on the valve turning task at DRC

Prepares queries for the motion planner by localizing the objects (e.g., valves) using interactive markers, sends the trajectory to the robot after visualization and approval by the Captain. The GUI operator should have extensive experience with both the user interface and the motion planner; should the motion planner fail (for instance, when there is no valid IK solution), they will be best positioned to work around the error.

Robot process operator

In addition to starting and stopping the software running aboard the robot (i.e., control and data aggregation systems), the robot process operator monitors debugging information logged by the various software components running on the robot. This operator should understand the system software architecture and have experience operating the robot. Should errors onboard the robot occur, this operator will both be the first to discover them and the best-equipped to address them.

Walking operator

Commands and monitors walking execution. This operator must be extremely familiar with the walking control of the robot, so that they can execute movement commands as quickly and efficiently as possible.

Network monitoring operator

In the DRC Trials, network communication with the robot can degrade dramatically. Thus it is useful to have an operator monitor the current network conditions. For instance, this operator can help the captain in his/her decisions to request sensor data or change sensor data rates and quality, which can overload the data link if made in a period of particularly poor network conditions. The network monitoring operator uses standard network diagnostics tools (for example, **ping**) to monitor network quality. The additional role of this operator, once again specific to the DRC Trials, is to serve as an on-field representative of the operating team during event setup and interventions.[3] This operator should have extensive test experience with the robot, so that they can assist in decision making both among the operators and on the field with the robot.

Because each operator's mental load is reduced using this cooperative control approach, adopting

[3] Interventions are five minute time-outs in which the robot can be serviced in person by the operators. A limited number (three) of interventions can happen in each task, either called for explicitly by the operators, or automatically triggered by the robot falling.

these roles enhances the robustness of the control process. Each operator is only responsible for a small number of tasks and the critical operations are monitored by at least two operators (i.e., the Captain and the operator responsible for the action). We also make use of a communication protocol where the name of the target operator is called prior to communicating to reduce the risk of mis-communication. For instance, in the valve-turning task, when the captain asks if the robot computers have received the trajectory, stating the robot process operator's name avoids ambiguity in the request which could unnecessarily load the other operators. This protocol is especially important in the distracting and stressful DRC Trials environment.

Reducing the cognitive load of each operator enables acting in a safer and more effective manner. While these tasks can be completed by lone operators – as we have done multiple times – in our experience, single operators are both slower and usually unable to quickly recover from errors. We believe team operation of a humanoid robot is especially efficient in scenarios where the robot must act under tight time constraints. Considerable precedent for this team structure exists; very similar operational modes are commonly found when operating large vehicles such as tanks or aircraft [11].

5 Testing

Preliminary versions of our system were tested on both the PR2 and Hubo2+ robots [2]. While these tests were suitable for debugging and evaluating the performance of various framework components, this testing was insufficient to discover all errors and limitations in our system. Moreover, it was not suitable for training the human operators; we dedicated little time to understanding how to recover from errors or how to expedite task completion. With preliminary development complete, we developed a testing schedule designed explicitly to test system performance and prepare operators for the DRC Trials. We believe similar methods could be used when developing systems and training operators for disaster response.

5.1 Testing Process

Our testing process consisted of a series of scheduled remote testing sessions. Remote tests were conducted over a VPN connection between our development team in Worcester, Massachusetts, USA and the robot at Drexel University in Philadelphia, Pennsylvania, USA. Remote testing time was split between sessions used to test newly-developed or modified features and "mock trial" sessions testing whole-system performance and the skills of the operators. These mock trials served as training tools for the operators, as they were representative of conditions encountered during the DRC Trials: no ability to observe the robot, limited communication with team members physically managing the robot, and degraded network conditions between the robot and operators. Though we did not store information on the quality of the

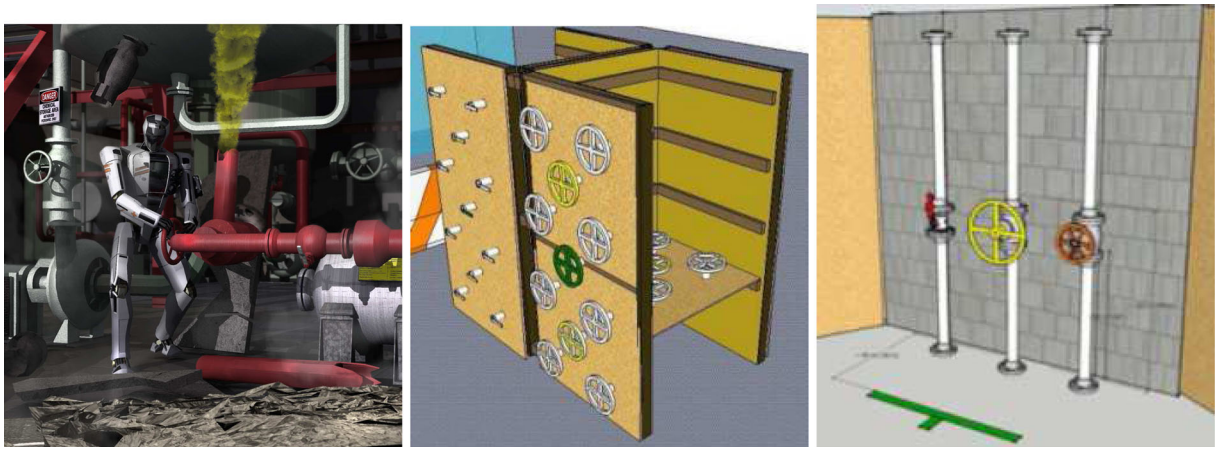

Fig. 7 Evolution of the task specs. provided by DARPA: **a** initial unstructed environment, **b** second specification with horizontal and vertical valve placements at different heights, **c** final task description with only vertically placed valves at a single height

network connection used for remote testing, we frequently observed latencies greater than 1 second and packet loss greater than 5 % – worse than the network conditions specified by DARPA.

The task specifications for the valve turning task are shown in Fig. 7, and the physical setup of our tests are shown in Fig. 8, with the robot standing in front of the valve. In all but the last remote test, the robot was already in position in front of the valve, while for the final remote test, the robot first walked up to the valves.

5.2 Testing Results

Remote tests were conducted over the course of three months, comprising eleven separate testing sessions, in which we performed the valve turning task in an environment similar to Fig. 8. Like the DRC Trials, each test was divided in two phases: setup time (15 minutes) and run time (30 minutes). Depending on the time taken by setup, we ran between one and three trials per session. The average setup time, run time and results are reported in Fig. 9.

Since our testing began before the DRC Trials' scoring rubric was published, we defined our own scoring criteria for our tests. However, since the task specification and score rubric published by DARPA changed multiple times our scoring rubric did not match the one used at the DRC trials. In the first phase of the tests we focused on turning a large circular valve three full turns. DRC rules were changed later to require only one turn on three valves in the setting presented in Fig. 7c. Our rubric was:

- 1 point - Grasping the valve
- 2 points - One full turn
- 3 points - Three full turns

Figure 9 reports the evolution of the average setup and run times in minutes, as well as the average scores. During the first test sessions, our code was unstable and startup was largely manual, as evidenced by the high setup and run time and the low average points. However, after the first two test sessions the average number of points per run was 2.26, setup time was 21.6 minutes and run time was 24.92 minutes. Those results indicate that we were able to perform over one turn of the valve at each trial and shows the overall reliability of the framework. On test seven we could not score points due to hardware failure in the setup increasing our setup-time and preventing us from completing a valid run.

On the last run (i.e., test 11, which is not reported in Fig. 9) we integrated the walking component from KAIST and adopted the final scoring rubric provided by DARPA. We aimed to turn all three different valves in a setting similar to Fig. 7c. On that test run the operational protocol as well as the software framework were the same as those used in the challenge. We scored four points according to DARPA's rubric by turning each valve a full turn within 30 minutes and not requiring any interventions.

Additional successful tests were conducted at the DRC Trials venue prior to competition. During the competition run we succeeded in turning the lever valve with the left hand. However, after completing a successful turn of the large round valve and releasing it, the robot lost balance and fell forward. The successful turning of the valve suggests that our approach for localizing the valve and turning it were effective, despite the limited compliance of the robot. While we cannot determine precisely the cause for the robot falling (a similar error did not occur in our previous tests), we believe the robot fell as a result of calibration error combined with insufficient balance control and the unexpected slope of the ground.

6 Lessons Learned

The system presented in Section 4 is the result of design and testing cycles in which significant tradeoffs have been made to maximize its performance on the valve-turning task.

The different iterations of the DRC rules regarding robot–workstation communications and mockup specifications acted as a moving target to our development. Initially, the vague requirements for the task, shown in Fig. 7a, encouraged us to implement a more autonomous approach. However, as the requirements became more precisely defined (e.g., no obstacles, only horizontal valves, single height for valve placements) and as automated techniques proved to be less effective than human operators, we moved back from autonomy towards a traded cooperative control approach.

In this section, we discuss our experience and lessons learned addressing each of the challenges discussed in Section 3. A common thread among these lessons is a shift from autonomy to teleoperation and,

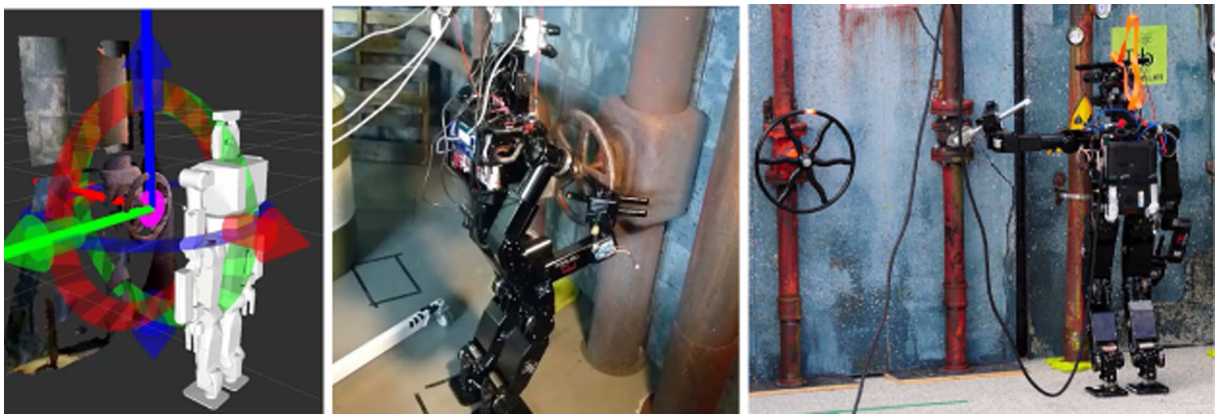

Fig. 8 DRCHubo performing the task: testing in a mockup environment at Drexel with **a** the operator's view in our user interface and **b** the robot turning the valve, and **c** the robot at the DRC Trials

with it, an increasing role for the human operator(s). We then discuss broader lessons learned implementing and using the cooperative traded control approach in relation to other teleoperation approaches.

6.1 Perception

A high level of autonomy in perception implies automatic detection and localization of objects in the scene. We avoided object detection because we did not have an appearance model of the object (i.e., color, the exact shape), however localization in a pointcloud can be performed by the Iterative Closest Point (ICP) algorithm when given a good initial guess by the operator and a parametrized model. ICP "snaps" the points on the surface of the object to the nearby points in the pointcloud. In fact, our preliminary system design incorporated ICP in the user interface [2].

Despite our initial use of ICP, the approach presented in Section 4.3 relies on the operator for both detection and localization of objects by manually

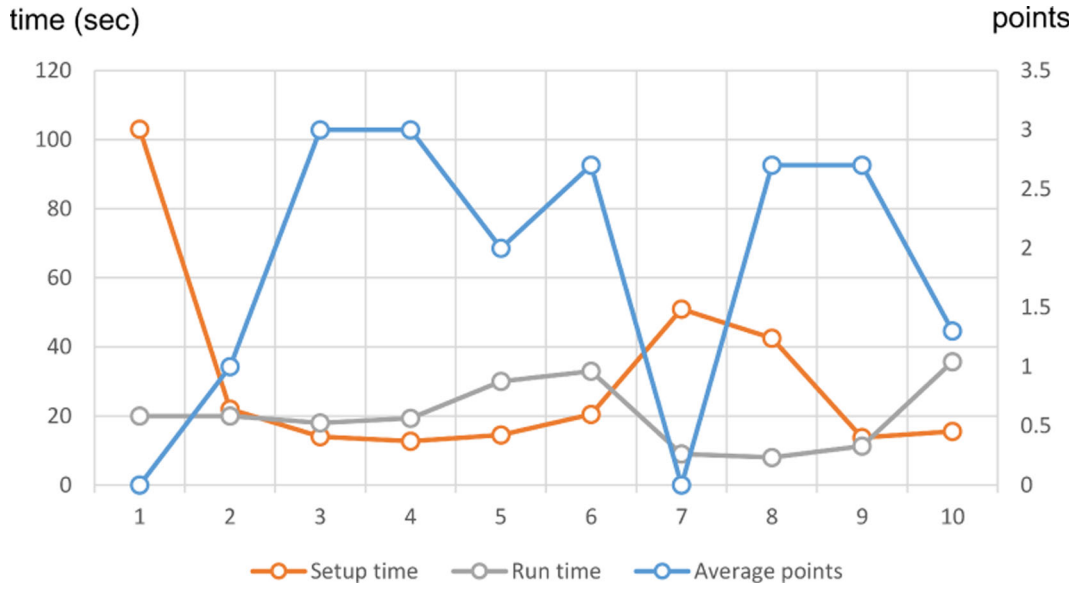

Fig. 9 Average setup time and run time in minutes (*left*) and average of points scored over 10 testing sessions (*right*). A test session comprise one to three tests

aligning shapes to the pointcloud data. In practice, operator localization of an object without using ICP was found to produce faster and more accurate results than using ICP. Indeed, ICP could find the object pose quickly when given a good initial guess [2], however, due to the sparsity of the data the operator often needed to provide several initial guesses, making the process slower than specifying a precise pose directly.

Summary Human-assisted perception can be significantly faster and more reliable than automated or semi-automated perception when used with experienced operators.

6.2 Base Placement

In unstructured environments, it is crucial to account for the robot's manipulation capabilities when selecting the placement location to perform a manipulation task. Initially, we pursued an autonomous solution to that problem based on reachability maps [52] to compute foot placements and configurations suitable for completing the task. Using a kinematic capability map of the robot, "promising" end-effector poses could be estimated. Using these poses, a set of foot placements could be computed. If a valid set of robot configurations could be found for both estimated end-effector poses and foot placements, then a suitable placement would be found.

However, as the DRC Trials rules developed, it became clear that the valve task environment would be highly structured, with few obstacles to complicate foot placement (see Fig. 7). We found that a skilled human operator was able to determine a successful placement faster than the autonomous algorithm in such a structured environment. In this case, a simple estimate of distance to the valve from the pointcloud data using an interactive marker by the operator was satisfactory. To assist the operators in making this selection, we tested a range of potential placements using our motion planner. From these tests we were able to determine distance ranges that would lead to successful valve turning.

Summary In simple structured environments, base placement selection for humanoid robots can be efficiently performed by experienced operators rather than by generating placements autonomously.

6.3 Manipulation

Our motion planner, CBiRRT, is able to account for obstacles as well as kinematic and balance constraints and thus can produce statically-stable motions. It has been very effective throughout our testing. In particular, when combined with our user interface, it is extremely good at encapsulating the complexities of humanoid manipulation. However, the algorithm does not account for uncertainty introduced by imperfect sensing. This uncertainty in valve position and obstacle locations can result in unexpected collisions or the hands of the robot becoming stuck on the valve. In the initial set of tests, sensing errors caused the robot to fall, as the robot collided with the valve when performing the End operation. While methods exist to account for uncertainty in sampling-based planning, we found that a simple solution based on adding way points in the reaching and extraction trajectories was sufficient. These way points are placed before grasping and after releasing the manipulated object, and the object's volume is augmented when planning motion to and from these way points. This procedure guarantees that the arms keep a minimal safety distance with the object to be manipulated as they perform the Ready and End motions. We found this solution effective enough to avoid collisions with the valves at all times.

Execution of planned trajectories is, alone, insufficient to confirm task completion. For the valve, a range of conditions, such as the hands slipping, missing the valve entirely due to sensing error, or being unable to turn the valve could prevent the task from being completed. Errors in task execution (i.e., when the task is not performed as intended) can be identified by using the dynamic programming technique Dynamic Time Warping (DTW) to match executed trajectories against a library of known successful and unsuccessful trajectories [2]. DTW iteratively calculates the best alignment between elements of two or more time sequenced data [41] and produces a metric that quantitatively represents the similarity of those sequences to either the successful or unsuccessful class, which facilitates error detection during execution. This technique gave reasonable results in testing with the PR2 (correct identification rate of 88 %). However, even this performance can be easily surpassed by an experienced human operator watching camera images of the task. Once it became clear that

a constant camera feed, albeit at low frequency and quality, could be maintained throughout the task, we switched to manual monitoring of execution.

Summary An effective motion planner is important to relieve the operators from the complexities of high-DoF manipulation. However, monitoring the status of task execution was more reliably performed by experienced human operators than by automated approaches.

6.4 Operation

Our operating protocol, with its division of work between multiple operators and clear chain of command, was effective at maintaining situation awareness and task performance. As noted before, our system was designed to be usable by a single operator if necessary; in fact, a significant number of remote tests were conducted partly or entirely by a single operator. However, single-person operation was marked by the difficulty of maintaining situation awareness while also completing the task quickly. In particular, recovering from errors was time-consuming and complicated for a single operator needing to monitor the state of the task, the robot, the robot's onboard software, and their user interface all at once.

Our switch to multiple operators, in which overall situation awareness is explicitly tasked to the "captain" separately from the robot operators, greatly improved operation. The combined knowledge and experience of the operating team allowed for more effective and informed decision making. Multiple operators working together and confirming each other's commands effectively eliminated the simple user errors that were experienced early in testing, and recovery from errors was made significantly more efficient with operators each managing a separate part of the system. An additional, unforeseen, benefit of the greater situation awareness provided by the multiple operator system was that the task could be completed with significantly lower quality sensor data. Compression artifacts and noise in sensor data that would otherwise have increased single-operator workload and chance for error instead posed minimal additional difficulty to the multi-operator team able to review the data and discuss any uncertainty together. Taking advantage of this allowed us to decrease data quality but increase the rate at which data (especially pointclouds) could be requested.

Summary Switching from a singe operator to a multi-operator team allowed for better management of situation awareness and reduced operator errors. In addition, multiple operators allowed for effective operation and situation awareness with less sensor data.

6.5 Communication

Specifications for the communications link between the robot and operator workstation varied between DRC kickoff and DRC Trials. Early rules incorporated bandwidth use into trials scoring, while the final rules only specified bandwidth and latency limits. Initial concerns about these rules drove much of our development focus towards autonomy and the initial development of the data link toolkit (see Section 4.2). As the task became more structured, our broader switch to operator-guided teleoperation increased the demands on the data link, since the operators were now required to complete tasks that previously would have been completed aboard the robot.

Accommodating this increase in data requirements was made possible by the development of our data link toolkit, which made possible the continuous transmission of images from the robot and the use of pointclouds when needed. This additional data greatly increased the situation awareness of the human operators. In our drive to reduce bandwidth demands, we discovered that experienced operators could reliably complete the task with lower-quality data (i.e., lower resolution, more compression artifacts, more noise) than had been feasible with autonomous solutions. The development of an extensive generic toolkit using ROS, rather than a special-purpose data link specific to the DRC, incurred additional up-front development time but resulted in considerably easier testing and implementation. Since all software could be tested with or without using the data link, its development could proceed in parallel to broader system development. This flexibility also makes our toolkit applicable to a wide range of robot use cases, including both disaster-recovery operations (ex. communications between robots and operators or other robots in disaster zones) and general robotics development (ex. mobile robots operating inside buildings with unreliable WiFi) in addition to its use in the DRC.

Summary The development of an effective toolkit for reducing bandwidth use provided for good situation awareness using camera images and pointclouds while remaining under tight bandwidth limits, and allowed us to offload perception tasks to human operators.

6.6 Testing

Our testing process, composed of a set of intensive on-site tests and extensive remote tests, was successful both in testing our system and preparing the operators for the DRC Trials. The reliance on remote testing forced the rapid development of our data link toolkit and prepared operators for the task with conditions representative of those encountered in the trials. In particular, the combination of unscored and scored remote tests allowed the development of new features, the testing of those features, and the evaluation of those features in terms of expected trial performance. A major lesson learned from the testing process was the inclusion of tests dedicated to the *use* of the system, rather than just system performance. These tests served to train the operators and identify usability issues that might otherwise have been ignored in favor of new feature development – for example, changes to our user interface that improved the speed and reduced the risk of error when commanding the planner.

Summary Testing of complex robotic systems should focus both on the correctness and performance of the system, but also the use of the system by its human operators. This provides experience and training for the users and identifies potential usability issues.

6.7 Teleoperation Method

The teleoperation method presented in Section 4 is an instance of *cooperative traded control* [18], which relies on an intermediate level of autonomy of the robot. We also experimented with a *direct control* approach as a fallback system for our framework, which requires less autonomy (i.e. substituting the motion planner for direct operator control of end-effector pose from the GUI). As expected, this teleoperation mode was significantly slower to use than the control mode presented in Section 4. From discussions with other teams competing in the DRC, it was indeed possible to use direct control for valve-turning, along with stored end-effector trajectories (e.g., a one-handed circle with a tool inserted into the valve spokes) to execute the turning motions (this was possible because the specifications for the valve locations were released not long before the competition). To perform the task with this strategy, the robot needs good balance control and arm strength to overcome the mismatch between the stored trajectory and the true valve pose and dimensions – which was not the case of DRCHubo using our system.

Another way to control the robot is by using *collaborative control* (similar to high-level supervision), in which the robot maintains more autonomy in task completion. Though this was our initial aim, our experience suggests that critical components in such a system may be less reliable or slower than direct human operation. As a result, it is unclear if state-of-the-art techniques would enable a collaborative control system to outperform a cooperative traded control approach.

Summary An intermediate level of autonomy, in which the robot manages low level details, such as the generation and execution of trajectories, while the human operators handle perception and overall task-level behavior, is an effective combination of the complementary strengths of robots and human operators.

7 Conclusions

We have presented a manipulation framework which applied to the valve-turning task of the DARPA Robotics Challenge. Our framework consists of a software framework for teleoperating humanoid robots to perform manipulation tasks with limited communication, an operating protocol designed to improve task completion and efficiency, and a testing process that simultaneously tested software and trained operators. Our testing process, consisting of hands-on intensive testing, remote testing, and remote practice runs, demonstrates that our framework is able to perform reliably and is resilient to unreliable network conditions. We emphasize that our system, while applied specifically to the valve-tuning task, can be applied to a variety of teleoperated robotic tasks – multiple components, with little or no modification, were used to complete other DRC tasks on our DRC team. We have analyzed our approach and discussed lessons

learned designing our system. In particular, we discuss several lessons which resulted in a system-wide shift from autonomy to cooperative traded control and show how those changes resulted in better performance than our original design. Our experience designing and using this system suggests that the key to building an efficient teleoperation system is to identify where a well-trained operator can surpass the performance of software components to get the best of both autonomy and human capabilities.

Acknowledgments This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) award #N65236-12-1-1005 for the DARPA Robotics Challenge. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

1. Almetwally, I., Mallem, M.: Real-time tele-operation and tele-walking of humanoid robot nao using kinect depth camera. In: 2013 10th IEEE International Conference on Networking, Sensing and Control (ICNSC), pp. 463–466 (2013). doi:10.1109/ICNSC.2013.6548783
2. Alunni, N., Phillips-Grafftin, C., Suay, H.B., Lofaro, D., Berenson, D., Chernova, S., Lindeman, R.W., Oh, P.: Toward a user-guided manipulation framework for high-DOF robots with limited communication. In: 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), pp. 1–6 (2013). doi:10.1109/TePRA.2013.6556356
3. Arkin, R.C., Balch, T.: Aura: principles and practice in review. J. Exp. Theor. Artif. Intell. **9**(2–3), 175–189 (1997)
4. Balakirsky, S., Carpin, S., Kleiner, A., Lewis, M., Visser, A., Wang, J., Ziparo, V.A.: Towards heterogeneous robot teams for disaster mitigation: results and performance metrics from RoboCup rescue. J. Field Rob. **24**(11–12), 943–967 (2007)
5. Balasubramanian, R., Xu, L., Brook, P.D., Smith, J.R., Matsuoka, Y.: Human-guided grasp measures improve grasp robustness on physical robot. In: 2010 IEEE International Conference on Robotics and Automation (ICRA), pp. 2294–2301 (2010). doi:10.1109/ROBOT.2010.5509855
6. Berenson, D., Srinivasa, S., Kuffner, J.: Task space regions: a framework for pose-constrained manipulation planning. Int. J. Robot. Res. (IJRR) **30**(12), 1435–1460 (2011)
7. Biesiadecki, J., Leger, C., Maimone, M.: Tradeoffs between directed and autonomous driving on the mars exploration rovers. In: Robotics Research, STAR, vol. 26, pp. 91–104. Springer, Berlin (2007)
8. Burridge, R.R., Hambuchen, K.A.: Using prediction to enhance remote robot supervision across time delay. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, IROS 2009, pp. 5628–5634 (2009). doi:10.1109/IROS.2009.5354233
9. Casper, J., Murphy, R.R.: Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. IEEE Trans. Syst. Man Cybern. B Cybern. **33**(3), 367–385 (2003)
10. Chitta, S., Jones, E.G., Ciocarlie, M., Hsiao, K.: Perception, planning, and execution for mobile manipulation in unstructured environments. IEEE Robotics and Automation Magazine, Special Issue on Mobile Manipulation (2012)
11. Cooke, G.W.: U.s. army tank doctrine (2004)
12. Dang, H., Jun, Y., Oh, P.K., Allen, P.: Planning complex physical tasks for disaster response with a humanoid robot. In: 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), pp. 1–6 (2013). doi:10.1109/TePRA.2013.6556365
13. Dantam, N., Stilman, M.: Robust and efficient communication for real-time multi-process robot software. In: 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 316–322 (2012). doi:10.1109/HUMANOIDS.2012.6651538
14. Diankov, R., Kuffner, J.: Openrave: a planning architecture for autonomous robotics. Robotics Institute, Pittsburgh, Tech Rep CMU-RI-TR-08-34. p. 79 (2008)
15. Diankov, R., Ratliff, N., Ferguson, D., Srinivasa, S., Kuffner, J.: BiSpace planning: concurrent multi-space exploration. In: Robotics: Science and Systems (2008)
16. Diftler, M.A., Mehling, J.S., Abdallah, M.E., Radford, N.A., Bridgwater, L.B., Sanders, A.M., Askew, R.S., Linn, D.M., Yamokoski, J.D., Permenter, F.A., et al.: Robonaut 2 - the first humanoid robot in space. In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 2178–2183 (2011). doi:10.1109/ICRA.2011.5979830
17. Ferrell, W., Sheridan, T.: Supervisory control of remote manipulation. IEEE Spectrum, pp. 81–88 (1967)
18. Goodrich, M.A., Crandall, J.W., Barakova, E.: Teleoperation and beyond for assistive humanoid robots. Reviews of Human Factors and Ergonomics **9**(1), 175–226 (2013)
19. Gossow, D., Leeper, A., Hershberger, D., Ciocarlie, M.: Interactive markers: 3-d user interfaces for ros applications. IEEE Robot. Autom. Mag. **18**(4), 14–15 (2011)
20. Grey, M.X., Dantam, N., Lofaro, D.M., Bobick, A., Egerstedt, M., Oh, P., Stilman, M.: Multi-process control software for HUBO2 Plus robot. In: 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), pp. 1–6 (2013). doi:10.1109/TePRA.2013.6556374
21. Hobbelen, D., de Boer, T., Wisse, M.: System overview of bipedal robots flame and tulip: tailor-made for limit cycle walking. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008, pp. 2486–2491 (2008). doi:10.1109/IROS.2008.4650728
22. Hornung, A., Bennewitz, M.: Adaptive level-of-detail planning for efficient humanoid navigation. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 997–1002 (2012). doi:10.1109/ICRA.2012.6224898
23. Hornung, A., Phillips, M., Jones, E.G., Bennewitz, M., Likhachev, M., Chitta, S.: Navigation in three-dimensional cluttered environments for mobile manipulation. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 423–429 (2012). doi:10.1109/ICRA.2012.6225029

24. Jackson, J.: Microsoft robotics studio: a technical introduction. IEEE Robot. Autom. Mag. **14**(4), 82–87 (2007)
25. Kragic, D., Miller, A.T., Allen, P.K.: Real-time tracking meets online grasp planning. In: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation, 2001, vol. 3, pp. 2460–2465 (2001). doi:10.1109/ROBOT.2001.932992
26. Lavalle, S.M.: Rapidly-exploring random trees: A new tool for path planning. Tech. rep. (1998)
27. Lum, M.J.H., Rosen, J., King, H., Friedman, D.C.W., Lendvay, T.S., Wright, A.S., Sinanan, M.N., Hannaford, B.: Teleoperation in surgical robotics – network latency effects on surgical performance. In: Annual International Conference of the IEEE, Engineering in Medicine and Biology Society, 2009. EMBC 2009, pp. 6860–6863 (2009). doi:10.1109/IEMBS.2009.5333120
28. Luo, J., Zhang, Y., Hauser, K., Park, H.A., Paldhe, M., Lee, C.S.G., Grey, M., Stilman, M., Oh, J.H., Lee, J., Kim, I., Oh, P.: Robust ladder-climbing with a humanoid robot with application to the DARPA robotics challenge. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 2792–2798 (2014). doi:10.1109/ICRA.2014.6907259
29. Medeiros, A.A.D.: A survey of control architectures for autonomous mobile robots. J. Braz. Comput. Soc. **4**(3) (1998)
30. Miller, A., Allen, P.: Graspit! a versatile simulator for robotic grasping. IEEE Robot. Autom. Mag. **11**(4), 110–122 (2004)
31. Oestges, C., Montenegro-Villacieros, B., Vanhoenacker-Janvier, D.: Modeling propagation into collapsed buildings for radio-localization-based rescue search missions. In: Antennas and Propagation Society International Symposium, 2009. APSURSI '09, pp. 1–4. IEEE (2009)
32. O'Flaherty, R., Vieira, P., Grey, M.X., Oh, P., Bobick, A., Egerstedt, M., Stilman, M.: Humanoid robot teleoperation for tasks with power tools. In: 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), pp. 1–6 (2013). doi:10.1109/TePRA.2013.6556362
33. Phillips-Grafflin, C.: Unreliable Network Communication Toolkit. https://github.com/WPI-ARC/teleop_toolkit (2013)
34. Pirjanian, P., Huntsberger, T.L., Trebi-ollennu, A., Aghazarian, H., Das, H., Joshi, S.S., Schenker, P.S.: CAMPOUT: a control architecture for multi-robot planetary outposts. In: Proceedings SPIE Conference Sensor Fusion and Decentralized Control in Robotic Systems III, pp. 221–230 (2000)
35. Pordel, M., Hellstrom, T.: Robotics architecture frameworks, available tools and further requirements. UMINF (2013)
36. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T.B., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
37. Rasmussen, C., Yuvraj, K., Vallett, R., Sohn, K., Oh, P.: Towards functional labeling of utility vehicle point clouds for humanoid driving. In: 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), pp. 1–6 (2013). doi:10.1109/TePRA.2013.6556368
38. Rusu, R.B., Cousins, S.: 3D is here: point cloud library (PCL). In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–4 (2011). doi:10.1109/ICRA.2011.5980567
39. Sakamoto, D., Kanda, T., Ono, T., Ishiguro, H., Hagita, N.: Android as a telecommunication medium with a human-like presence. In: 2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 193–200 (2007)
40. Sarabia, M., Ros, R., Demiris, Y.: Towards an open-source social middleware for humanoid robots. In: Humanoids (2011)
41. Senin, P.: Dynamic time warping algorithm review. Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA (2008)
42. Shannon, C.E.: A Mathematical Theory of Communication. Bell System Technical Journal **27**, 379–423, 623–656 (1948)
43. Sheridan, T.B.: Telerobotics, Automation and Human Supervisory Control. MIT Press, Cambridge (1992)
44. Srinivasa, S., Ferguson, D., Helfrich, C., Berenson, D., Collet, A., Diankov, R., Gallagher, G., Hollinger, G., Kuffner, J.: VandeWeghe M. HERB: a home exploring robotic butler. Autonomous Robots (2009)
45. Stilman, M.: Task constrained motion planning in robot joint space. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007. IROS 2007, pp. 3074–3081 (2007). doi:10.1109/IROS.2007.4399305
46. Stulp, F., Fedrizzi, A., Mösenlechner, L., Beetz, M.: Learning and reasoning with action-related places for robust mobile manipulation. J. Artif. Intell. Res. **1**, 1–42 (2012)
47. Tadokoro, S.: Special project on development of advanced robots for disaster response (DDT project). In: IEEE Workshop on Advanced Robotics and its Social Impacts (2005)
48. (TTO) DRCTTO: Darpa robotics challenge. Accessed 16 Jan 2013 (2012)
49. Yakey, J., LaValle, S.M., Kavraki, L.E.: Randomized path planning for linkages with closed kinematics chains. IEEE Trans. Robot. Autom. **17**(6), 951–959 (2001)
50. Yanco, H.A., Drury, J.L., Scholtz, J.: Beyond usability evaluation: analysis of human-robot interaction at a major robotics competition. Hum. Comput. Interact. **19**(1–2), 117–149 (2004)
51. Zacharias, F., Borst, C., Beetz, M., Hirzinger, G.: Positioning mobile manipulators to perform constrained linear trajectories. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008, pp. 2578–2584 (2008). doi:10.1109/IROS.2008.4650617
52. Zacharias, F., Sepp, W., Borst, C., Hirzinger, G.: Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories. In: 9th IEEE-RAS International Conference on Humanoid Robots, 2009. Humanoids 2009, pp. 55–61 (2009). doi:10.1109/ICHR.2009.5379601
53. Zheng, Y.F., Wang, H., Li, S., Liu, Y., Orin, D., Sohn, K., Jun, Y., Oh, P.: Humanoid robots walking on grass, sands and rocks. In: 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), pp. 1–6 (2013). doi:10.1109/TePRA.2013.6556367

54. Zucker, M., Jun, Y., Killen, B., Kim, T.-G., Oh, P.: Continuous trajectory optimization for autonomous humanoid door opening. In: 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), pp. 1–5 (2013). doi:10.1109/TePRA.2013.6556358

Calder Phillips-Grafflin is a Ph.D. student and Research Assistant in the Robotics Engineering program at Worcester Polytechnic Institute (WPI). In 2012, he received a BS in Computer Engineering from Union College and joined the Autonomous Robotic Collaboration (ARC) Lab at WPI. His current research interests are focused on motion planning and manipulation with deformable objects.

Halit Bener Suay is a Ph.D. student and a Research Assistant at Worcester Polytechnic Institute (WPI), Robotics Engineering Program. In 2011, he joined Robot Autonomy and Interactive Learning (RAIL) group. He has a B.Sc. and a M.Sc. degree in Aeronautics Engineering from Istanbul Technical University and The University of Tokyo. His current research interest is focused on Learning from Demonstration for robots.

Jim Mainprice is a Postdoctoral fellow in the Autonomous Motion Department at the Max Planck Institute for Intelligent Systems (Tubingen, Germany) since January 2015. He received his M.S. from Polytech' Montpellier, France, and his Ph.D. in robotics and computer science from the University of Toulouse, France, in 2009 and 2012 respectively. His research interests include motion planning, task planning, machine learning, human-robot collaboration and human-robot interaction.

Nicholas Alunni received a MS in Robotics Engineering from Worcester Polytechnic Institute in 2013. He currently works at Amazon Robotics as a Software Engineer.

Daniel M. Lofaro is an Assistant Professor in the Electrical and Computer Engineering Department at George Mason University. He is the director of Lofaro Labs Robotics LLC and the Drones and Autonomous Systems Lab @George Mason University (DASL@GMU). An NSF-EAPSI and ONR-SFRP Fellow, he received his doctorate from the ECE Department at Drexel University in 2013. From 2012-2013 he was the Research Lead of the DARPA Robotic Challenge team DRC-Hubo. His research interests include Complex Control Systems and Robotics with most recent ventures relating to Robot Design and Cloud Robotics.

Dmitry Berenson received a BS in Electrical Engineering from Cornell University in 2005 and received his Ph.D. degree from the Robotics Institute at Carnegie Mellon University in 2011. He completed a post-doc at UC Berkeley in 2011 and started as an Assistant Professor in Robotics Engineering and Computer Science at WPI in 2012. He founded and directs the Autonomous Robotic Collaboration (ARC) Lab at WPI, which focuses on motion planning, manipulation, and human-robot collaboration.

Sonia Chernova is an Assistant Professor of Computer Science and Robotics Engineering at Worcester Polytechnic Institute and the director of the Robot Autonomy and Interactive Learning (RAIL) lab. She received her Ph.D. degree in Computer Science from Carnegie Mellon University in 2009, and worked as a Postdoctoral Associate at the MIT Media Lab prior to joining WPI. Her research interests span robotics, interactive machine learning, adjustable autonomy, human computation and human-robot interaction. Dr. Chernova's research is supported through funding from NSF, ONR and DARPA, including the NSF CAREER and ONR YIP awards. Her work has been covered by national and international press, including the New York Times, National Geographic, NPR and the BBC.

Dr. Robert W. Lindeman is an Associate Professor in the Department of Computer Science at Worcester Polytechnic Institute (WPI) in Massachusetts, USA. Rob is Director of the Human Interaction in Virtual Environments (HIVE) Lab, which focuses on immersive, multi-sensorial feedback systems for VR, AR, and gaming, as well as natural and non-fatiguing interaction. He is Director of the WPI's Interactive Media & Game Development (IMGD) Program. Rob received his MS degree from the University of Southern California in 1992, and his Doctor of Science (ScD) degree from The George Washington University in 1999 in the areas of Computer Graphics and Human-Computer Interaction. Rob is a Senior Member of both the IEEE and ACM.

Prof. Paul Oh is the Lincy Professor of Umanned Aerial Systems at the University of Nevada, Las Vegas (UNLV). Prior to joining UNLV, he was with Drexel University's Mechanical Engineering Department from 2000–2014 where he founded and directed the Drexel Autonomous Systems Lab. He received mechanical engineering degrees from McGill (B.Eng 1989), Seoul National (M.Sc 1992), and Columbia (PhD 1999) universities. Honors include faculty fellowships at NASA Jet Propulsion Lab (2002), Naval Research Lab (2003), the NSF CAREER award (2004), the SAE Ralph Teetor Award for Engineering Education Excellence (2005) and being named a Boeing Welliver Fellow (2006). He is also the Founding Chair of the IEEE Technical Committee on Aerial Robotics and UAVs. From 2008-2010, he served at the National Science Foundation (NSF) as the Program Director managing the robotics research portfolio. He has authored over 100 referred archival papers and edited 3 books in the areas of robotics and unmanned systems. He serves as Editor for several leading robotics publications including Springer-Verlag's Journal of Intelligent Robotics and Systems and the Journal of Intelligent Service Robotics. He also served as Director for the NATO Advanced Studies Institute (ASI) in 2010 on Unmanned Systems which gathered researchers from over 20 countries to capture the state of the art and formulate research roadmaps. In 2012, he served as Program Chair for the flagship conference for the academic robotics community, the IEEE International Conference on Robotics and Automation (ICRA), held in St. Paul, MN, USA. For the DARPA Robotics Challenge, he served as lead for Team DRC-Hubo (2012–2013) and Team DRC-Hubo@UNLV (2014–2015). In recognition of his international partnerships and impact on US research and education, he was one of three Distinguished Lecturers invited by the National Science Board to speak at their 60th Anniversary in 2010.