Intel Serv Robotics (2014) 7:121–131
DOI 10.1007/s11370-014-0156-8

SPECIAL ISSUE

Toward a user-guided manipulation framework for high-DOF robots with limited communication

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

Received: 8 August 2013 / Accepted: 7 May 2014 / Published online: 2 July 2014

Abstract This paper presents our progress toward a user-guided manipulation framework for high degree-of-freedom robots operating in environments with limited communication. The system we propose consists of three components: (1) a user-guided perception interface that assists the user in providing task-level commands to the robot, (2) planning algorithms that autonomously generate robot motion while obeying relevant constraints, and (3) a trajectory execution and monitoring system which detects errors in execution. We report quantitative experiments performed on these three components and qualitative experiments of the entire pipeline with the PR2 robot turning a valve for the DARPA robotics challenge. We also describe how the framework was ported to the Hubo2+ robot with minimal changes which demonstrates its applicability to different types of robots.

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

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

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

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

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

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

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

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

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

Keywords Humanoid robotics · Manipulation · Motion planning · Teleoperation · Robot software architecture

1 Introduction

We seek to create a user-guided manipulation framework for high degree-of-freedom (DoF) robots such as humanoids and mobile manipulators operating in environments with limited communication. Application of our framework to these robots is conducive to greater autonomy and enables tasks ranging from home maintenance and care for the elderly or disabled to disaster response in conditions that are hazardous to humans. While a great deal of research has explored methods for perception [12], error-recovery [9], motion planning [12,27,28], and teleoperation [11,30] for such applications, our goal was to unify existing algorithms in a reliable general-purpose manipulation framework.

This paper presents our progress toward such a framework. We evaluate our framework by performing valve turning, which is one of the tasks required for the DARPA robotics challenge (DRC) [17]. For a description of our DRC team's work on other tasks, see [15,22,33,37,43,54,55]. The valve-turning task requires that a robot locate, approach, grasp, and turn an industrial valve with two hands and thus presents a challenging test case for our system due to the perception and dexterous manipulation involved. Another core constraint is the limited communications with the robot, making conventional teleoperation infeasible.

Hence, the valve-turning task requires a straightforward way for a user to command the robot to perform complex actions. These actions require accurate localization of the valve relative to the robot, constrained motion planning for closed-chain kinematic systems, and autonomous error detection to report problems to the user. Thus, the system we propose consists of three main parts: (1) a user-guided perception interface which provides task-level commands to the robot, (2) a planning algorithm that autonomously generates robot motion while obeying relevant constraints, and (3) a trajectory execution and monitoring system which detects errors in execution. Our goal was that all three of these parts be usable on different robots in both the physical world and simulated environments.

In the first component, a user roughly aligns a model of the relevant object (e.g., a valve) to a pointcloud provided by the robot's sensors. While autonomous perception algorithms have previously been applied for such tasks, they have difficulty with highly unstructured environments and underspecified tasks like those encountered in the DRC. However, given a good guess for the location of an object, these algorithms can be quite effective. Thus, we use the iterative closest point (ICP) algorithm [6] to "snap" a rough user-generated alignment into place. Once satisfied with the alignment, the user commands the robot to perform the task.

The manipulation planning component of the system consists of the CBiRRT algorithm [4], which is capable of generating constrained quasi-static motion for high-DoF robots. Once a trajectory is constructed by the planning component, it is executed by the execution monitoring component. The monitoring component compares the execution of the current trajectory to a library of previous executions of the same task (generated from previous runs) to detect errors. This component uses dynamic time warping (DTW) [48] to compute an error metric between trajectory executions.

We have found that our user interface has a success rate of 97 % of finding the object's real-world location, when given a good user guess at the object's position relative to the robot. The planning algorithm we used successfully generated feasible object manipulation trajectories under constraints 94 % of the time within 25 s, and our trajectory execution error detector correctly identified whether or not the valve was successfully turned in 88 % of trajectories on the PR2 robot.[1]

The rest of the paper is structured as follows: Sect. 2 gives a background on the relevant technologies and topics and Sect. 3 describes the system architecture and components. Section 4 shows the quantitative analysis of our framework when applied on the PR2 robot and Sect. 5 shows the preliminary results on the Hubo2+ robot. Section 6 describes the extension of our framework to the Hubo2+ humanoid robot. In Sect. 7, we discuss future work and finally Sect. 8 concludes the paper.

[1] A video of the framework in operation can be seen at: http://www.youtube.com/watch?v=xRcUO2mXt3s.

2 Background

2.1 Service-oriented architectures

A service-oriented architecture (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 [41]. There are many options for SOAs currently available today, including Microsoft Robot Design Studio (MRDS), Joint Architecture for Unmanned Systems (JAUS), Hierarchical Attentive Multiple Models for Execution and Recognition (HAMMER) [46], and Robot Operating System (ROS) [42].

ROS was chosen for this work due to its extensive proven ability on the PR2, a high-DoF robot with two arms and a mobile base. ROS has also been applied to more anthropomorphic humanoid robots such as the Nao [1], Robonaut 2 [19], and TU/e TUlip [25]. Additionally, ROS was chosen for its built-in visualization tool (RVIZ), which allows for fast user interface development, integration with a simulator (Gazebo) [29] for both physics and sensor simulation, and a large repository of open-source code. For more information on robotic frameworks, both free and commercially available, please see [14,24]. To the best of our knowledge, there is no available framework for high-DoF robots, unlike UAVs or rovers, that is tailored for user-guided object manipulation in unstructured environments with limited connection to the robot.

2.2 Low-bandwidth communication

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, 2–10, and greater than 10 s latency [8]. For example, many systems that operate between 0 and 2 s of latency are surgical systems. Such systems can even operate across distant locations [32]. Latency greater than 2 s is typically found in research related to earth orbit or farther systems, such as Lunar robots [8,40]. Latency greater than 10 s extends even farther including the Mars rovers, which have a delay of many minutes [7]. Low bandwidth can also necessitate dynamically assigning resources between robots in order to function in the limited communication environment [45].

Highly unstructured disaster environments provide a challenge where communication can be difficult due to the

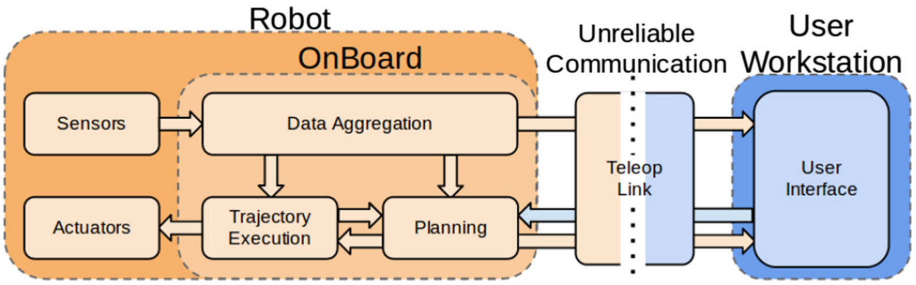


Fig. 1 System diagram showing data flow through the framework

unknown properties of the building materials, making transmission and reception of signals unreliable [36]. The framework we create is meant to operate with a latency between 0 and 5 s, with packet loss that will be analogous to communication in a demolished building. To address the shortcoming of ROS when used over degraded networks, we have created datalink software designed to ensure reliable communication with the robot. Additionally, in order to compensate for the poor communication link, our datalink software allows the user to dynamically adjust the bandwidth used by each sensor's data.

2.3 Robot teleoperation

There have also been many architectures that attempt to tackle the problem of manipulation performed by a robot with a mobile base [26]. The problems present in these mobile manipulation tasks include where to place the robot in relation to the object to be manipulated [51], how to grasp the object [35], and how to plan the robot's movements [4]. This framework is designed for a mobile robot manipulating an object whose general shape, but not size and location, is known a priori. Due to these unknowns, the framework must be able to place the robot in a location, where the object can be manipulated, grasp locations must be determined, and trajectories must be generated dynamically for an object whose size and pose are unknown before being specified by a user.

High-level supervision [20] has become a popular way of controlling mobile robots compared with conventional teleoperation or full autonomy. This shift has happened due to a variety of reasons, including the difficulty of the perception problem, issues with navigating in an unstructured environment, and problems that arise when attempting to teleoperate a high-DoF system such as a humanoid [8]. In high-level supervision, the robot performs autonomous actions that are specified by a user to overcome these challenges. The role of the user has therefore shifted from an operator, who dictates every movement, to a supervisor, who guides at a high-level [49]. This approach is often used for unmanned systems controlled from a central command post [3,23,34]. The user will act as a supervisor when using our framework by specifying the pose of an object to be manipulated. Traditional teleoperation would be challenging for the tasks we consider due to the many DoF to control and the latency and packet loss when communicating with the robot.

3 Architecture

Our framework, shown in Fig. 1, is implemented using ROS [42] for communication with the robot and the user interface, and OpenRAVE [18] for motion planning. The framework consists of a set of modules that provide data aggregation, user interface, motion planning, and trajectory execution. Below we describe each module in Fig. 1.

3.1 Data aggregation

The primary function of the data aggregation package is to format data coming from the robot. The data aggregation package takes in sensor data, which varies depending on the robot, and re-publishes it in a standard format so that the framework can be easily implemented on a variety of robots. As shown in Fig. 1, data aggregation is the only component of the framework that receives data such as pointclouds and values from accelerometers and encoders directly from the robot's sensors. This design allows the system to be highly modular and quickly adapted to different robots, including switching between robots operating in real and simulated environments. If necessary, this component can be reconfigured during operation to handle changes in the available sensor data, such as changing which pointcloud sensor to use throughout the system.

This package also provides synthesized information, such as collision maps and object proximity, that is derived from raw sensor data. This information synthesis is performed onboard the robot to reduce the need for communication. For instance, depth images generated from downsampled pointclouds reduce the data transmission need by nearly 90 %. For more details on the implementation and performance of the data aggregation system, see [39].

3.2 User interface

Due to the difficulty of autonomous perception, we created a graphical user interface (GUI) to aid in the detection of

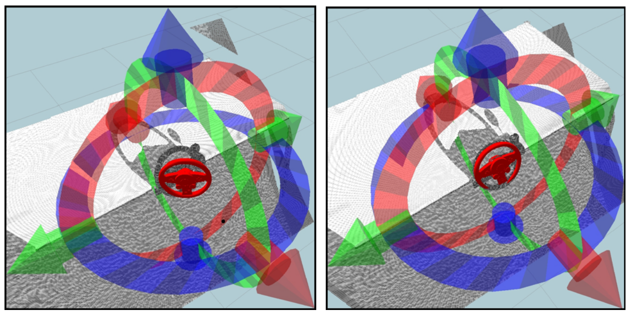


Fig. 2 Iterative closest point being used to align an object in RVIZ. **a** The object before ICP has been run, **b** the final translation after ICP has finished

objects. Using the GUI, as shown in Fig. 2a, the user manipulates an interactive marker [21] to hint at an object's location. The interactive marker displayed to the user can be a rectangle, disk, or a triangle mesh. Object alignment is then performed using the ICP algorithm which minimizes the error between two specified groups of points. ICP is a standard algorithm for computing the alignment between 3D shapes, and an easy-to-use implementation is provided in PCL [44]. ICP "snaps" a given input to the target world, as shown in Fig. 2b, by iteratively computing the best alignment between points on the surface of the object model and nearby points in the pointcloud. To decrease computation time, we use a bounding box to extract a subset of the pointcloud that is near the user's guess.

In addition to user input and feedback, the GUI controls data flow over the unreliable link to the robot. Data from the robot is transmitted only when specifically requested to minimize communication. This architecture takes advantage of the assumption that the robot inhabits a largely static environment, such as the DRC's valve-turning task.

3.3 Motion planning

The motion planning component generates collision-free manipulation trajectories for high-DoF robots which respect balance and end-effector pose constraints. The initial placement of the robot is critical to the resulting motion. Indeed, the robot must be able to reach and manipulate the object for the entirety of the task while maintaining balance, avoiding self-collisions, and collisions with the environment. For each manipulation task, initial and goal configurations are first computed using inverse kinematics.

The second phase consists of planning trajectories between the current configuration and the inverse kinematic solutions. This step is performed by the CBiRRT algorithm [4], 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 [50,53], 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 implicitly defined by the constraints.

3.4 Trajectory execution

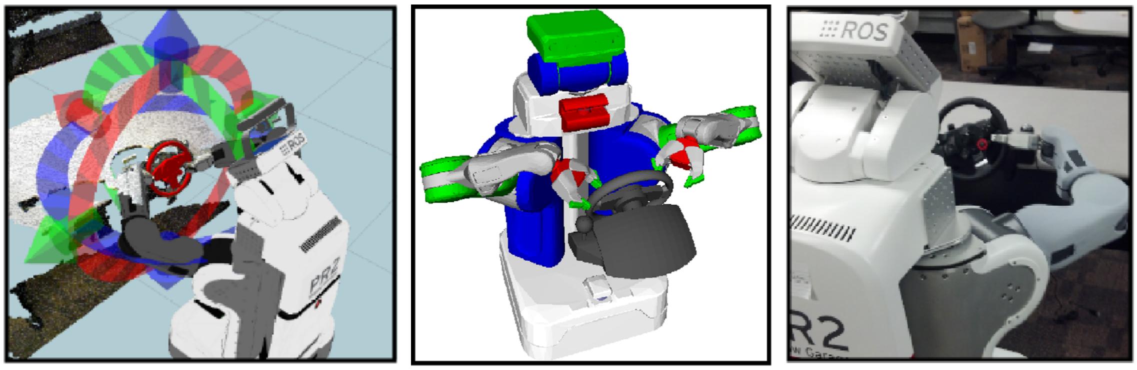
The trajectory execution package executes a planned trajectory and detects errors encountered during execution. For this error detection, trajectories are recorded during execution using only the data available from joint encoders. No other contextual data, such as the output from the planner or the pose of the object being manipulated, is required.

Errors in trajectory execution are identified by using the dynamic programming technique Dynamic Time Warping (DTW) to match executed trajectories against a library of known successful and unsuccessful trajectories. DTW iteratively calculates the best alignment between elements of two or more time sequenced data [48] and produces a metric that quantitatively represents the similarity of those sequences to each other. DTW has previously been applied to the problem of assessing similarity between trajectories [5]. In our framework, DTW provides a distance metric between end-effector pose trajectories, allowing the trajectory execution system to find the closest trajectory in the library. To account for trajectories significantly different from those in the library, cases in which the computed DTW metric is greater than an experimentally determined threshold can be automatically identified as error conditions.

This method of error detection using DTW is well suited to our task as it requires no complex visual feedback and no special sensors and is thus applicable to a wide range of robots using only the data already available from basic joint encoders. In particular, this method is appropriate for the DRC where it may not be possible to determine the state of the valve through other means. However, the method may not generalize well to tasks where a library of executions would be difficult to generate (e.g., if the shape of the object is completely unknown). For our testing, we used this approach to determine whether the valve manipulated by the PR2 was successfully turned.

3.5 Teleoperation datalink

To support operations in environments with limited bandwidth and unreliable networks, we have developed a set of ROS tools specifically for communications in networks with



(a) PR2 in RViz **(b)** PR2 in OpenRAVE **(c)** The real PR2

Fig. 3 The PR2 Robot as seen in **a** the RVIZ visualization engine performing valve alignment, **b** OpenRAVE for motion planning, and **c** the real world performing valve turning

low bandwidth, dropouts, and high latency. Our teleoperation link package consists of a number of tools based on those provided by the *topic_tools* package built into ROS, with several improvements and extensions to enable higher reliability, simpler configuration, and use in single-master and multi-master systems. Our teleoperation datalink software is available as an open-source ROS package [39].

In particular, the primary components of our teleoperation datalink system are relays and rate controllers that allow fine-grained control over the network demands of our system. A set of generic relays, suitable for all ROS message types, is provided to handle relaying data over an unreliable network link. These relays extend the simple equivalents provided in ROS with automatic detection of network problems, notification and warnings to the user, and automatic recovery of broken network sockets. These relays enable reliable ROS datalinks that are robust to network dropouts.

Our teleoperation link software provides both a set of rate-limiting repeaters and a ROS service API for rate control. Using the rate-limiting repeaters allows for data published at a "native" rate of 200 Hz onboard the robot to be forwarded to the user's workstation at a low rate of 10–20 Hz, reducing network demands and data usage. Additionally, the same repeaters provide for single-message requests for data that will not be streamed continuously. The service API provided allows for the integration of rate and request control within other software, such as sensor drivers, to remove the need for a dedicated rate-control node. An RVIZ plugin in our user interface, shown in Fig. 7, provides a simple GUI for a user to modify the rate at which the sensor data are transmitted.

4 Framework validation

The framework we have developed allows for a user to hint at the location of an object in the world and have the robot approach and manipulate the object. In order to perform this action, we must determine the pose of the object, generate a trajectory to manipulate it from a start configuration, and monitor the trajectory for errors during execution. We performed quantitative experiments on the aforementioned components of the architecture separately, and qualitative tests of the entire framework using a Willow Garage PR2 performing the valve-turning task both in simulation and in the real world. The PR2 is a large mobile manipulator with two seven-DoF arms, a sensor head, and a holonomic mobile base. The robot is equipped with cameras, depth sensors, and an IMU. The main stages of our framework using the PR2 are shown in Fig. 3.

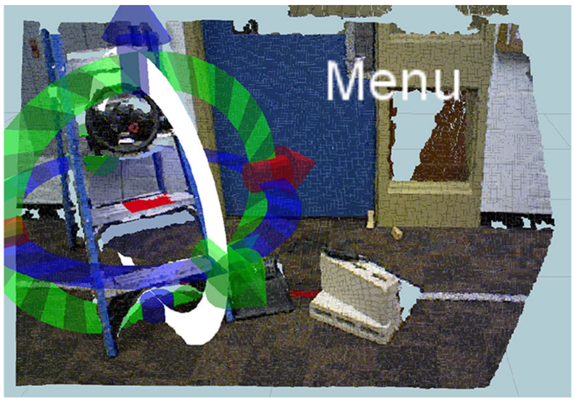


Fig. 4 Successful alignment of a rectangular interactive marker to a ladder rung using ICP

4.1 Object alignment

To enable semi-automated testing of ICP object alignment, the user interface provides an option to automatically generate "noisy" object alignments. We used this automated tester to evaluate the valve alignment system by randomly perturbing the valve's position. The "error" of each simulated user guess, E is calculated by adding the total translation offset,

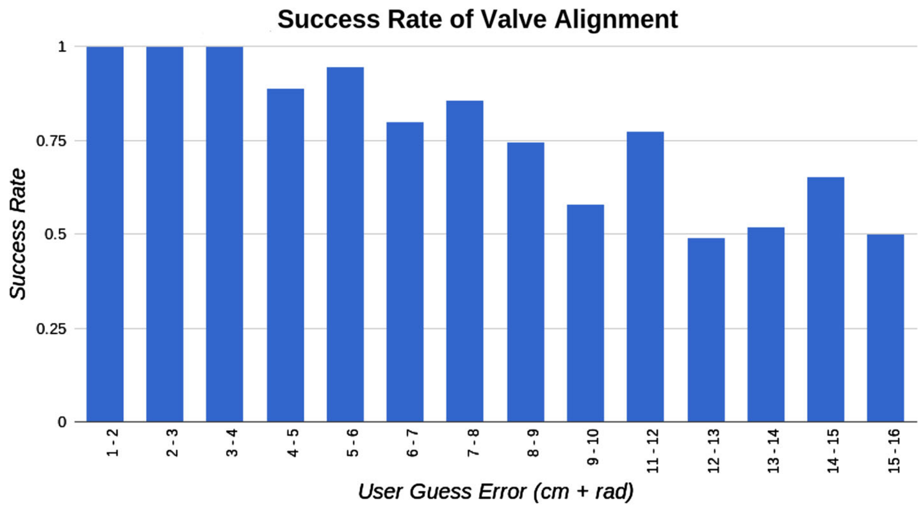


Fig. 5 Success rate (final error less than 0.3 units) of ICP to find the actual valve position given a randomly generated perturbation in the value's position

E_t, to the total rotational offset, E_q. The true position of the valve is denoted as $V_o = [X_o, Y_o, Z_o]$ and the guessed position of the valve is denoted as $V_g = [X_g, Y_g, Z_g]$. The total translation value E_t is calculated as the Euclidean distance, in cm, between the two frame origins. Each valve's pose also contains a quaternion of the form (x, y, z, w) that represents its orientation in space. The difference in angle between the quaternion representing the valve's position and the guessed position, E_q, is calculated using the equation below:

$$E_q = \arccos(2*((x_o*x_g)+(y_o*y_g)+(z_o*z_g)+(w_o*w_g))^2-1) \tag{1}$$

Figure 5 shows the success rate of 450 sample alignments with random perturbations, where success is defined as a final error of less than 0.3 units. The "user guess error" is the amount of error introduced by the automated tester. We qualitatively categorized the 450 sample alignments: one to five error units was considered a "good" user guess, six to ten error units was considered an "acceptable" user guess, and eleven to fifteen error units was a "poor" user guess. ICP was capable of producing a successful alignment 97 % of the time for a good guess, 79 % of the time for an acceptable guess, and 58 % of the time for a poor guess. The number of ICP iterations, 500, remained constant throughout all tests. This number of iterations was determined so that ICP returns with a new valve alignment in under 1 s; however, the number of iterations can be increased, allowing larger transformations to be found at the cost of longer runtime.

To further test the alignment system, we created a scene with different items that would require object alignment for manipulation. These items included a ladder, valve, cinder block, cutting tool, and a door with a handle. The interactive marker can be a rectangle, a disk, or a user-specified mesh to better represent different objects in the scene, and aligning to those objects was tested using ICP. Figure 4 shows the ICP output of an alignment to a ladder rung using the system.

4.2 Motion planning

We tested the CBiRRT trajectory planner's performance with the constraints described in Sect. 3 with a range of valve poses. For each test run, we defined six pose constraints and two path constraints in the CBiRRT framework. For the valve-turning task, we designed a path that consists of four trajectories:

1. From initial pose (*init*) to valve grasping configuration (*start*),
2. From valve grasping configuration (*start*) to the configuration right after turning the valve 45 degrees clockwise (*turned*),
3. From *turned* back to *start*, and
4. From *start* back to *init*.

Init, start, and turned configurations are shown in Fig. 6.

Starting with a known successful valve pose, we generated random valve poses with rotational offsets ranging from zero and 85 degrees and translational offsets between 0 and 0.1 m. These randomly generated poses were first validated by the inverse kinematics (IK) solver to ensure that the start and end configurations necessary for the robot were feasible; 422 poses were confirmed by the IK solver and used for testing. Of these 422 poses, the trajectory planner succeeded in planning all four trajectories 94 % of the time. The planner was given a maximum time of 25 s to plan each of the four trajectories.

4.3 Trajectory execution

To test the trajectory execution system, we generated a library of known valid and invalid trajectories. Valid trajectories

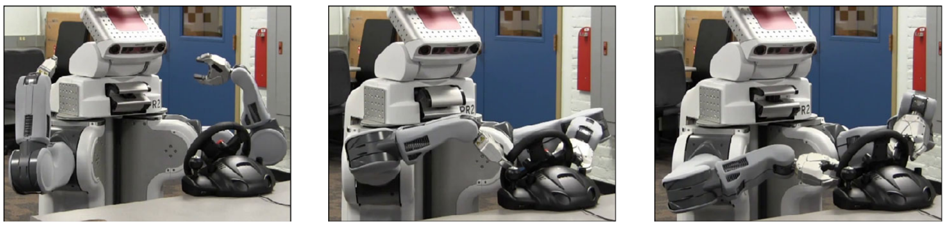


Fig. 6 Configurations for valve-turning path

(a) Initial Configuration **(b)** Start Configuration **(c)** Turned Configuration

were those in which both hands maintained a grasp on the valve throughout the entire trajectory, and invalid trajectories were those in which either hand failed to grasp or lost grasp of the valve in mid-turn.

We manually created an initial library of 22 known valid and invalid example trajectories by providing correct and incorrect alignments of the valve to the planner and tagged the resulting trajectory as either successful or not. The library was then grown to 526 trajectories through further experiments. In these experiments, we introduced random rotational noise between 0 and 10 degrees and translational noise between 0 and 0.025 m. These values were selected to represent reasonable valve misalignments given the observed noise and error of the sensors on the PR2. Also note that our data compression system induces errors no greater than 0.02 m. The executions of these trajectories were tagged by hand. This process could be accelerated using a ground-truth metric (e.g., an encoder that measured the true position of the wheel) to provide automatic tagging of examples.

In our framework, DTW is sufficiently fast to enable online evaluation of individual trajectories. Each evaluation against the 526-element library is completed in less than 4 s, whereas execution of the actual trajectories requires approximately 32 s. We performed leave-one-out testing of the library itself, in which each trajectory in the library was compared against the other trajectories in the library to test the ability of the error detector to correctly identify if an error was encountered. This testing resulted in an overall correct identification rate of 88 %. Of the trajectories identified as successful, 10 % of these trajectories were actually unsuccessful. Of the trajectories identified as unsuccessful, 16 % of them were actually successful. This discrepancy between identification rates is acceptable because the cost potentially incurred from a false-negative identification (which would result in a retry of the alignment and planning) is significantly lower than that from a false positive (which would result in prematurely terminating the valve-turning task).

4.4 Communications link testing

The communications link serves two distinct functions: limiting the bandwidth used to send sensor data from the robot to the operator(s) and ensuring that the communication between robot and operator(s) is maintained despite degraded network conditions. These degraded network conditions include low bandwidth (down to 100 Kbit/s), high latency (up to 1 s), packet loss, and periodic dropouts.

In order to test the bandwidth limiting capabilities, data from different sensors on the robot was streamed through the link at their default rate and downsampled to a rate specified by a user. The streamed sensor data included data from joint encoders, force-torque sensors, tilt sensors, and an RGB camera. Table 1 shows the bandwidth used by the data of different sensors after they have been downsampled by the communication link. The rate at which the sensor data is transmitted can be increased or decreased through a RVIZ plugin shown in Fig. 7.

To test the resiliency of our communications link software, we simulated network conditions using the Dummynet

Table 1 Data rate and bandwidth of sensors after the communication link

Sensor	Rate (Hz)	Bandwidth (Kb/s)
Joint states	20	210
Force/torque sensors	20	60
IMU and tilt sensors	20	161
Camera color—compressed	30	160
Camera B & W—compressed	30	100

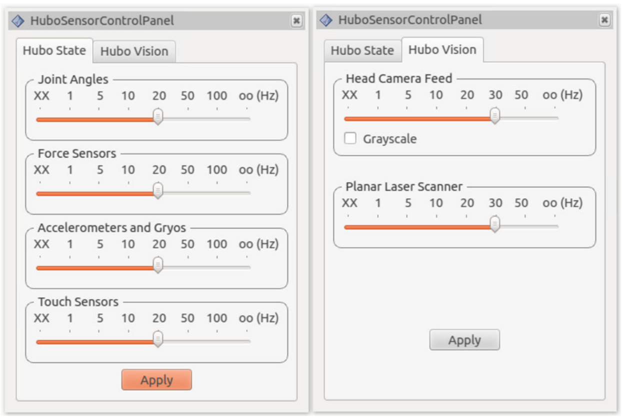


Fig. 7 The RVIZ plugin that allows the user to control the data rate of the robot's sensors through the graphical user interface

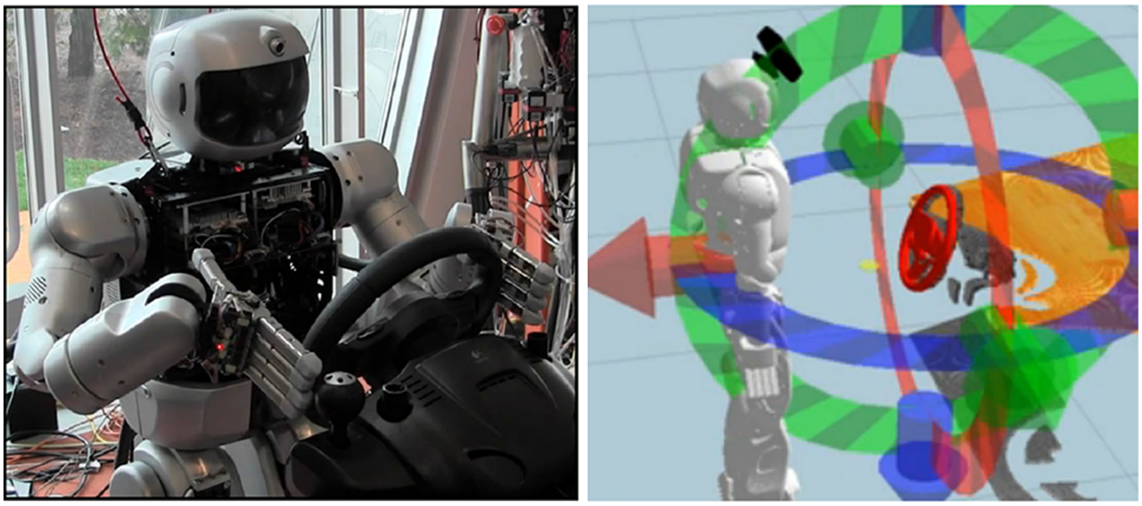


Fig. 8 The Hubo2+ robot turning a valve (**a**) and shown in RViz (**b**)

traffic shaping tool [10]. Our communications link successfully transmitted camera images in conditions ranging from a best case of 1 Gbit/s with minimal latency (less than 1 ms) to a worst case of 50 Kbit/s with latency of 5 s, packet loss of 25 %, and periodic dropouts of up to 20 s. In these network conditions, our communications link significantly outperforms the tools built into ROS; at 1 Mbit/s our software provides approximately one-and-a-half times as many images per second, at 100 Kbit/s two times as many, and at 50 Kbit/s three times as many. As expected from these tests, our communications software performed reliably at the DRC Trials, where bandwidth was limited between 1 Mbit/s and 100 Kbit/s with latency of 100 ms to 1 s.

4.5 Full framework testing

In order to test the operation of the entire framework, we ran 20 complete test cycles. Each test cycle consisted of three major steps: (1) we manually drove the PR2 to a random location in the room from where it could see the valve, (2) an expert user identified the location of the valve using the GUI and sent the location of the valve to the planning component, and (3) the PR2 approached the valve autonomously, planned valve-turning trajectories, and then executed those trajectories. For these tests, we created a valve analog from a commercially available force-feedback racing wheel.

The average time to run the entire framework from start to finish was approximately two minutes. On average, turning toward the valve and driving to a location where it could be manipulated took 30 s, planning took 2–3 s, and turning the valve took approximately 90 s. Of the 90 s spent turning the valve, trajectory execution accounted for 64 s, trajectory classification with DTW took 12 s, and closing and opening the grippers accounted for the remainder.

Observed performance of the trajectory execution system on the PR2 shows that false-negative identifications correlate with “borderline” trajectories. These trajectories, due to a combination of compliance in the PR2’s arm joints and inaccuracy in sensing, maintain a grasp on the valve in some executions and slip off in others. In practice, these borderline trajectories would be unable to turn a valve requiring significant torque to operate.

5 Hubo humanoid implementation

Hubo2+ is a 130 cm (4′3″) tall humanoid robot shown in Fig. 8a. It was designed and built by the Hubo Lab in the Korean Advanced Institute of Science and Technology (KAIST) [13,31]. The Hubo2+ has six DoF in each leg, six in each arm, five in each hand, three in the neck, and one in the waist, with a total of 38 DoF.

For our testing, we use Hubo-Ach, a real-time control daemon that uses a high-speed, low-latency IPC called Ach [16] to communicate with controllers. All of the joint controllers are independent processes and are able to be terminated at anytime without adversely affecting the Hubo-Ach daemon. Hubo-Ach implements a real-time loop in which all of the motor references and state data are set and updated, respectively. Communication with motor controllers is provided over CAN.

5.1 Framework implementation on Hubo2+

Our framework required a small number of changes in order to apply it to the Hubo robot. First, the data aggregation component, which is the only component to receive data directly from the robot itself, needed to be changed to topics specific to the Hubo robot instead of the PR2. The planning package was modified to account for a robot with balance constraints by adding a support polygon determined from Hubo's feet. Due to the difference in the two-handed workspace of the Hubo robot compared with the PR2, Hubo was placed 30 cm from the valve instead of 50 cm and turned the wheel 18 degrees instead of 32 degrees. Finally, the model displayed to the user in the GUI was updated. Figure 8b shows the Hubo robot in the RVIZ environment with the user-guided localization marker.

6 ROS–Ach Interface

To apply our framework written in ROS to the Hubo robot, we created a ROS package, *hubo_ros_core*, to support the Hubo robot. In particular, this interface is designed to abstract Hubo-specific implementation details and provide a common interface similar to that of other robots using ROS. Using this interface, the same software we have developed for the PR2 can be used on a radically different robot.

Hubo_ros_core provides several infrastructure components for working with the Hubo robot: common message types, a direct interface for publishing robot state and joint control, and a mid-level interface to execute actions on the robot.

A package of message types, *hubo_robot_msgs*, provides standard communication messages for everything from joint states to high-level actions such as pointing the head or requesting a LIDAR scan. These messages provide a common baseline for all ROS software running on the Hubo robot that allows them to share data and command actions on the robot.

Low-level support for the Hubo robot in ROS is provided by a bridge between ROS and Hubo-Ach, which provides interfaces to the Hubo-Ach real-time layer. The interface to Hubo-Ach provides both data *out* from the robot's sensors (joint encoders, force-torque, tilt, and IMU) and control *in* to directly control joint positions. Using the data read from the real-time layer, this package provides transformations between all link frames on the robot. This allows for transformation of data in one frame to another using existing tools in ROS. In addition, the transforms enable visualization of the robot in RVIZ, which forms the core of our user interface.

Support for mid-level execution, namely joint- and pose-space trajectories, is provided in a set of packages. These packages provide for trajectory execution using standard ROS interfaces that allow our planning software, developed for the PR2, to easily control the Hubo robot in a nearly identical manner. Other high-level actions, such as pointing the head of the robot to track a specific point or conducting a scan with the onboard sensors, are provided by these packages. The aim of these packages is to provide a common execution system for all higher-level actions and behavior implemented for the Hubo robot that mirror those available for other robots that support ROS. This commonality allows for the easy porting of existing software to the Hubo, near-identical interfaces between multiple Hubo variants, and a familiar interface to new users.

Our interface, *hubo_ros_core*, is available as open-source software and can be used with minor configuration changes on any variant of the Hubo robot that supports the Hubo-Ach real-time control layer [38].

We validated our system using the Hubo2+, as shown in Fig. 8a, successfully replicating the PR2's open-loop valve-turning ability.

7 Future work

Our framework provides significant room for expansion to the user interface, planning system, and trajectory execution monitor. The current interface provides neither a method of searching for an object nor autonomous object detection. Future versions will combine these with a variety of improvements to increase user situation awareness.

Our current motion planning system is dependent on the manual generation of both initial configurations and task constraints. Automatic generation of these is an important avenue for further work. We would like to develop an automated configuration predictor using human-agent knowledge transfer techniques that have been shown to be effective for teaching agents different types of tasks [2,52].

For trajectory execution, which is the final stage in our system, we would like to extend our implementation to identify different kinds of error conditions. We intend to improve the performance of the underlying DTW implementation using a variety of established [48] and novel techniques [47] to use not only larger trajectory libraries, but also to increase the resolution at which we evaluate the trajectories for errors.

8 Conclusion

In this paper, we have presented the foundation of a novel framework for user-guided manipulation with high degree-of-freedom robots in environments with limited communication. This framework includes techniques for object identification, constrained trajectory generation, and trajectory monitoring. We presented a quantitative evaluation of all major components in simulation, and qualitative experiments on the framework as a whole. We found that the system was effective at the valve-turning task on two very different robots.

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. We would also like to thank Russ Tedrake of CSAIL, MIT for allowing us to use his Hubo for testing.

References

1. Almetwally I, Mallem M (2013) Real-time tele-operation and tele-walking of humanoid robot nao using kinect depth camera. In: IEEE International Conference on Networking, Sensing and Control, pp 463–466
2. Argall BD, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robotics and Autonomous Systems 57(5):469–483
3. Arkin RC, Balch T (1997) AuRA: Principles and Practice in Review. Journal of Experimental and Theoretical Artificial Intelligence 9:175–189
4. Berenson D, Srinivasa S, Kuffner J (2011) Task Space Regions: A Framework for Pose-Constrained Manipulation Planning. International Journal of Robotics Research 30(12):1435–1460
5. Berenson D, Abbeel P, Goldberg K (2012) A robot path planning framework that learns from experience. In: IEEE International Conference on Robotics and Automation, pp 3671–3678, doi:10.1109/ICRA.2012.6224742
6. Besl P, McKay ND (1992) A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14(2):239–256
7. Biesiadecki J, Leger C, Maimone M (2007) Tradeoffs between directed and autonomous driving on the mars exploration rovers. In: Thrun S, Brooks R, Durrant-Whyte H (eds) Robotics Research, Springer Tracts in Advanced Robotics, vol 28. Springer, Berlin Heidelberg, pp 254–267
8. Burridge R, Hambuchen K (2009) Using prediction to enhance remote robot supervision across time delay. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 5628–5634
9. Butner S, Ghodoussi M (2003) Transforming a surgical robot for human telesurgery. IEEE Transactions on Robotics and Automation 19(5):818–824
10. Carbone M, Rizzo L (2010) Dummynet revisited. ACM SIGCOMM Computer Communication Review 40(2):12
11. Chen TL, Ciocarlie M, Cousins S, Grice P, Hawkins K, Hsiao K, Kemp CC, King CH, Lazewatsky DA, Leeper A, Nguyen H, Paepcke A, Pantofaru C, Smart WD, Takayama L (2012) Robots for humanity: User-centered design for assistive mobile manipulation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 5434–5435
12. Chitta S, Jones EG, Ciocarlie M, Hsiao K (2012) Perception, Planning, and Execution for Mobile Manipulation in Unstructured Environments. IEEE Robotics and Automation Magazine, Special Issue on Mobile Manipulation 19(2):58–71
13. Cho BK, Park SS, ho Oh J (2009) Controllers for running in the humanoid robot, HUBO. In: IEEE-RAS International Conference on Humanoid Robots
14. Craighead J, Murphy R, Burke J, Goldiez B (2007) A survey of commercial open source unmanned vehicle simulators. In: IEEE International Conference on Robotics and Automation, pp 852–857
15. Dang H, Jun Y, Oh P, Allen P (2013) Planning complex physical tasks for disaster response with a humanoid robot. In: IEEE International Conference on Technologies for Practical Robot Applications, pp 1–6
16. Dantam N, Stilman M (2012) Robust and efficient communication for real-time multi-process robot software. In: IEEE-RAS International Conference on Humanoid Robots
17. DARPA Tactical Technology Office (2012) DARPA Robotics Challenge Broad Agency Announcement. [Online; accessed 16-Jan-2013]
18. Diankov R, Kuffner J (2008) OpenRAVE: A planning architecture for autonomous robotics. Tech. Rep. CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania
19. Diftler M, Mehling J, Abdallah M, Radford N, Bridgwater L, Sanders AM, Askew RS, Linn D, Yamokoski J, Permenter F, Hargrave B, Piatt R, Savely R, Ambrose R (2011) Robonaut 2 - the first humanoid robot in space. In: IEEE International Conference on Robotics and Automation, pp 2178–2183
20. Ferrell W, Sheridan T (1967) Supervisory control of remote manipulation. IEEE Spectrum 4(10):81–88
21. Gossow D, Leeper A, Hershberger D, Ciocarlie M (2011) Interactive Markers: 3-D User Interfaces for ROS Applications. IEEE Robotics and Automation Magazine 18(4):14–15
22. Grey M, Dantam N, Lofaro D, Bobick A, Egerstedt M, Oh P, Stilman M (2013) Multi-process control software for HUBO2 Plus robot. In: IEEE International Conference on Technologies for Practical Robot Applications, pp 1–6
23. Hannaford B (1989) A design framework for teleoperators with kinesthetic feedback. IEEE Transactions on Robotics and Automation 5(4):426–434
24. Harris A, Conrad J (2011) Survey of popular robotics simulators, frameworks, and toolkits. In: IEEE Southeastcon, pp 243–249
25. Hobbelen D, de Boer T, Wisse M (2008) System overview of bipedal robots flame and tulip: Tailor-made for limit cycle walking. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2486–2491
26. Hornung A, Phillips M, Jones E, Bennewitz M, Likhachev M, Chitta S (2012) Navigation in three-dimensional cluttered environments for mobile manipulation. In: IEEE International Conference on Robotics and Automation, pp 423–429
27. Hsiao K, Ciocarlie M, Brook P (2011) Bayesian Grasp Planning. ICRA 2011 Workshop on Mobile Manipulation: Integrating Perception and Manipulation
28. Khatib O (1987) A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE Journal of Robotics and Automation 3(1):43–53
29. Koenig N, Howard A (2004) Design and use paradigms for Gazebo, an open-source multi-robot simulator. IEEE/RSJ International Conference on Intelligent Robots and Systems 3:2149–2154
30. Leeper A, Hsiao K, Ciocarlie M, Takayama L, Gossow D (2012) Strategies for human-in-the-loop robotic grasping. In: ACM/IEEE International Conference on Human-Robot Interaction, pp 1–8
31. Lofaro DM, Oh P (2012) Humanoid Throws Inaugural Pitch at Major League Baseball Game: Challenges, Approach, Implementation and Lessons Learned. In: IEEE/RSJ International Conference on Intelligent Robots and Systems
32. Lum M, Rosen J, King H, Friedman D, Lendvay T, Wright A, Sinanan M, Hannaford B (2009) Teleoperation in surgical robotics - network latency effects on surgical performance. In: International Conference of the IEEE Engineering in Medicine and Biology Society, pp 6860–6863
33. Luo J, Zhang Y, Hauser K, Park H, Paldhe M, Lee C, Grey M, Stilman M, Oh J, Lee J, Kim I, Oh P (2014) Robust ladder-climbing with a humanoid robot with application to the darpa robotics challenge. In: IEEE International Conference on Robotics and Automation (to appear)
34. Medeiros AAD (1998), A survey of control architectures for autonomous mobile robots. Journal of the Brazilian Computer Society 4
35. Miller A, Allen P (2004) Graspit! a versatile simulator for robotic grasping. IEEE Robotics and Automation Magazine 11(4):110–122

36. Oestges C, Montenegro-Villacieros B, Vanhoenacker-Janvier D (2009) Modeling propagation into collapsed buildings for radio-localization-based rescue search missions. In: IEEE Antennas and Propagation Society International Symposium, pp 1–4
37. O'Flaherty R, Vieira P, Grey M, Oh P, Bobick A, Egerstedt M, Stilman M (2013) Humanoid robot teleoperation for tasks with power tools. In: IEEE International Conference on Technologies for Practical Robot Applications, pp 1–6
38. Phillips-Grafflin C (2013) Hubo ROS Core. https://github.com/WPI-ARC/hubo_ros_core
39. Phillips-Grafflin C (2013) Unreliable Network Communication Toolkit. https://github.com/WPI-ARC/teleop_toolkit
40. Pirjanian P, Huntsberger T, Trebi-Ollennu A, Aghazarian H, Das H, Joshi S, Schenker P (2000) CAMPOUT: A control architecture for multi-robot planetary outposts. Proceedings of SPIE Symposium on Sensor Fusion and Decentralized Control in Robotic Systems III 4196:221–230
41. Pordel M, Hellstrom T (2013) Robotics architecture frameworks, available tools and further requirements. UMINF
42. Quigley M, Gerkey B, Conley K, Faust J, Foote T, Leibs J, Berger E, Wheeler R, Ng A (2009) ROS: an open-source Robot Operating System. In: ICRA workshop on open source software
43. Rasmussen C, Yuvraj K, Vallett R, Sohn K, Oh P (2013) Towards functional labeling of utility vehicle point clouds for humanoid driving. In: IEEE International Conference on Technologies for Practical Robot Applications, pp 1–6
44. Rusu RB, Cousins S (2011) 3D is here: Point Cloud Library (PCL). IEEE International Conference on Robotics and Automation. Shanghai, China, pp 1–4
45. Rybski P, Stoeter S, Gini M, Hougen D, Papanikolopoulos N (2001) Effects of limited bandwidth communications channels on the control of multiple robots. IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE 1:369–374
46. Sarabia M, Ros R, Demiris Y (2011) Towards an open-source social middleware for humanoid robots. In: IEEE-RAS International Conference on Humanoid Robots, pp 670–675
47. Sart D, Mueen A, Najjar W, Keogh E, Niennattrakul V (2010) Accelerating Dynamic Time Warping Subsequence Search with GPUs and FPGAs. In: IEEE International Conference on Data Mining, pp 1001–1006
48. Senin P (2008) Dynamic time warping algorithm review. Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA pp 1–23
49. Skrzypczyliski P (1997) Supervision and teleoperation system for an autonomous mobile robot. IEEE/RSJ International Conference on Intelligent Robots and Systems 2:1177–1181
50. Stilman M (2007) Task constrained motion planning in robot joint space. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp 3074–3081
51. Stulp F, Fedrizzi A, Mösenlechner L, Beetz M (2012) Learning and reasoning with action-related places for robust mobile manipulation. Artificial Intelligence Research 43(1):1–42
52. Taylor ME, Suay HB, Chernova S (2011) Integrating reinforcement learning with human demonstrations of varying ability. In: The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2
53. Yakey J, LaValle SM, Kavraki LE (2001) Randomized Path Planning for Linkages with Closed Kinematics Chains. IEEE Transactions on Robotics and Automation 17(6):951–959
54. Zheng Y, Wang H, Li S, Liu Y, Orin D, Sohn K, Jun Y, Oh P (2013) Humanoid robots walking on grass, sands and rocks. In: IEEE International Conference on Technologies for Practical Robot Applications, pp 1–6
55. Zucker M, Jun Y, Killen B, Kim TG, Oh P (2013) Continuous trajectory optimization for autonomous humanoid door opening. In: IEEE International Conference on Technologies for Practical Robot Applications, pp 1–5