An Architecture for Online Affordance-based Perception and Whole-body Planning

Maurice Fallon, Scott Kuindersma, Sisir Karumanchi, Matthew Antone, Toby Schneider, Hongkai Dai, Claudia Pérez D'Arpino, Robin Deits, Matt DiCicco, Dehann Fourie, Twan Koolen, Pat Marion, Michael Posa, Andrés Valenzuela, Kuan-Ting Yu, Julie Shah, Karl Iagnemma, Russ Tedrake, and Seth Teller
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar Street, Cambridge, Massachusetts 02139

Received 0 xxxx 0000; accepted 0 xxxx 0000

The DARPA Robotics Challenge Trials held in December 2013 provided a landmark demonstration of dexterous mobile robots executing a variety of tasks aided by a remote human operator using only data from the robot's sensor suite transmitted over a constrained, field-realistic communications link. We describe the design considerations, architecture, implementation, and performance of the software that Team MIT developed to command and control an Atlas humanoid robot. Our design emphasized human interaction with an efficient motion planner, where operators expressed desired robot actions in terms of affordances fit using perception and manipulated in a custom user interface. We highlight several important lessons we learned while developing our system on a highly compressed schedule.

1. INTRODUCTION

Since the start of the DARPA Robotics Challenge (DRC) in early 2012, we have worked as a team toward the realization of a software system for a human-scale robot that, given only limited communication with its human operators, can perform useful tasks in complex and unfamiliar environments—first in the simulation environment of the Virtual Robotics Challenge (VRC) in May 2013, and later in the physical DRC Trials. This paper describes the key ideas underlying our technical approach to the DRC, and the architecture that we adopted as we moved from conception to implementation. We elucidate several key components of our system—the human-robot interface; the perception, planning, and control modules; and the networking infrastructure—to illustrate how our approach met the requirements of the DRC Trials in December 2013. Finally, we look back on the previous two years, both to give a sense of the experience of competing, and to draw useful lessons for the final DRC competition and for the field.

There are several key ideas behind our approach. The first is that, since the system includes a human operator in communication with the robot, the robot need not be fully autonomous. We chose to allocate the traditional executive (high-level task planning) functions to the operator, along with the responsibility to partition high-level plans into smaller sequences for autonomous execution by the robot. We aimed to encompass sequences of varying complexity, from small motions, such as would be executed by a conventional teleoperated robot, through primitive actions such as stepping, reaching, grasping and lifting, and finally to subtasks such as walking, fetching, and using tools.

Our second key idea is to frame all operation of the robot in terms of operator-perceived *affordances*, or environmental elements that hold possibilities for action (Gibson, 1977). We adopted an interface metaphor in which the operator first helps the robot perceive and interpret a task-salient affordance in its surroundings, then commands the robot to act in terms of this conveyed affordance. For example, to turn a valve, the operator first coarsely indicates the valve wheel in the currently available sensor data; the system responds by fitting an adjustable valve wheel *template* to this datum. Once the operator is satisfied that the system has an actionable model of the valve wheel, s/he can command the robot to turn the valve. Templates can also incorporate valuable information provided by the system designers well in advance, such as manipulation stances or grasp locations likely to be successful. In this way, our system factors some of the challenging and expensive aspects of online motion planning into an offline phase.

The third key idea is to combine optimization with human input. For perception, a human operator provides the rough morphology, location, and attitude of each affordance, after which the system uses optimization to fit parameters more precisely to sensor data. In footstep planning, the operator provides a goal and a maximum number of footsteps, and an optimizer fills in suitable footsteps given

Direct correspondence to: mfallon@mit.edu

View this article online at wileyonlinelibrary.com • DOI: 10.1002/rob.21546

the terrain estimated by the perception system. In multi-contact planning, the human operator provides sequence points at which contacts are made and broken, mitigating the combinatorial complexity of algorithmic planning.

Beginning in March 2012, when we first heard that a new DARPA Challenge was being mounted, we gathered key people and system components. The lead PI (Teller) and colead (Tedrake) were joined by a perception lead (Fallon), a planning and control lead (Kuindersma), and a manipulation lead (Karumanchi). We incorporated two key software modules from previous projects: lightweight communications and marshaling (LCM) (Huang, Olson, & Moore, 2010), a lightweight message-passing infrastructure and tool set for visualizing and logging streaming sensor data, and Drake (Tedrake, 2014), a powerful MATLAB toolbox for planning, control, and analysis using nonlinear dynamical systems. LCM was developed as part of MIT's 2006-7 DARPA Urban Challenge (DUC) effort, and has since been released as open-source and used for a wide variety of projects both within and outside of MIT. Drake was initially developed by Tedrake and his research group to provide the planning and control infrastructure for numerous legged robot and unmanned aerial vehicle (UAV) projects; we have now released this code as open-source as well.

DARPA's timeline for the Challenge was highly compressed. Each team first had to choose the *track* in which to compete: Track A teams would develop both hardware and software, whereas Track B teams would develop software only, and upon successful qualification in a simulation environment, would receive a humanoid robot from DARPA. Excited by the opportunity to use a Boston Dynamics robot, we chose to compete as a Track B team.

Although our team brought considerable relevant experience to the project, we had very little direct exposure to humanoid robots. There were only eight months between the kick-off meeting in October 2012—where we received our first draft of the rules—and the Virtual Robotics Challenge in June 2013; we subsequently received our Atlas robot on August 12, 2013, and had to compete again, this time with the real robot, only four months later. In this brief period, we developed substantial components of our system from scratch (including, for instance, an optimization-compatible kinematics and dynamics engine, and a complete user interface) and integrated complex existing software. Many of us made considerable personal sacrifices for the project, devoting countless hours because we believed it was an incredible and unique opportunity.

1.1. Prior Work

Depending on how one defines the scope of the DRC, prior work can be interpreted very broadly—spanning all of humanoid, disaster response, and emergency robotics research.

Tracked robots such as iRobot's Packbot (Yamauchi, 2004) have been used to explore disaster situations, including the Fukushima Daiichi disaster (which motivated the DRC). Others, such as Tohoku University's Quince, have been paired with quad-rotors to demonstrate collaborative situational awareness (Michael et al., 2012), but broadly speaking robotics have been unable to make a major contribution beyond passively observing the scene.

Furthermore, the possibility of robots operating entirely autonomously and providing comparable capability to human-teleoperated systems seems remote indeed. Accepting these limitations, efforts have been made to standardize testing of robotic capability and to quantify progress (Sheh et al., 2014), which has in turn contributed to the definition of the DRC tasks. Additionally, field reports have benchmarked the usage of teleoperated robots after disasters such as the World Trade Center attack (Casper & Murphy, 2003) and the Tohoku earthquake (Matsuno et al., 2014).

Separately, humanoid research has been ongoing for some decades (Kemp et al., 2008). As the DRC progresses, participants have moved to build upon this research but also to provide an industrial strength manipulation capability absent in the previous generation of humanoids. Publications by DRC participants include Alunni et al. (2004), which described a system in which a human can remotely guide a Hubo robot to manipulate a valve. Dellin, Strabala, Haynes, Stager, & Srinivasa (2014) analyzed the performance of the CHIMP robot (CMU) at the DRC Trials in a manner similar to our own analysis (see Section 7), illustrating task execution that requires considerable time for fitting (or fixturing), planning, and teleoperation—which varies between tasks. They also noted that operation speed and success were correlated with operator training.

2. OVERVIEW OF THE ATLAS ROBOT

Atlas is a full-scale, hydraulically actuated humanoid robot manufactured by Boston Dynamics. The robot stands approximately 188 cm (6 ft 2 in) tall and has a mass of approximately 155 kg (341 lb) without hands attached. It has 28 actuated degrees of freedom (DOFs): six in each leg and arm, three in the back, and one neck joint. A tether attached to the robot supplies high-voltage three-phase power for the onboard electric hydraulic pump, distilled water for cooling, and a 10 Gbps fiber-optic line to support communication between the robot and field computer. Despite its considerable size, the robot is capable of moving quickly due to its ability to produce large forces (up to several hundred N-m in the leg joints) at high bandwidth.

Several sensor signals are available for integration with perception and control algorithms. Joint position, velocity, and force measurements are generated at 1000 Hz on the robot computer and transmitted back to the field computer at 333 Hz. Joint positions are reported by linear variable differential transformers (LVDTs) mounted on the actuators; velocities are computed through numerical differentiation of the position signals. Joint forces are estimated using pressure sensors in the actuators. In addition to the LVDT sensors, digital encoders mounted on the neck and arm joints give low-noise position and velocity measurements. A six-axis IMU mounted on the pelvis produces acceleration and rotation measurements used for state estimation. Two six-axis load cells are mounted to the wrists, and arrays of four strain gauges, one in each foot, provide three-axis force-torque sensing.

Atlas has four primary behavior modes: balancing, stepping, walking, and user mode. The balancing mode provides basic dynamic balance capabilities where the pelvis pose and upper-body DOFs are under operator control. The stepping behavior provides a basic walking capability that is driven by inputs specifying desired step locations and leg swing trajectories. The walking behavior is a faster and more dynamic locomotion mode, but because it is significantly less accurate and predictable than stepping, we did not use it in the DRC Trials. Finally, user mode puts all joint-level control under the command of the operator.

The Multisense SL sensor head was manufactured by Carnegie Robotics and consists of a stereo camera and a spindle-mounted Hokuyo UTM-30LX-EW planar LIDAR. Two Point Grey Blackfly cameras with 185-degree fisheye lenses were mounted on the upper torso for improved situational awareness. We also used a microphone located on the head to detect acoustic events (in particular drilling sounds).

2.1. Hands and End Effectors

We used a variety of end effectors that mounted directly onto the robot's wrist force sensors (Figure 1). The rules of the competition allowed us to change end effectors between tasks, so long as the robot carried all of its hands for all tasks. These included two types of actuated hands, the iRobot Dexter Hand and the Robotiq 3-Finger Adaptive Gripper. While the iRobot hand provided useful tactile sensing in its fingers and palm, the ruggedness of these sensors was unfortunately insufficient for our aggressive daily testing. We also used inert hooks and pointers in tasks that required large contact forces such as ladder climbing.

2.2. Kinematic Reachability and Sensor Coverage

The robot's arms have only six DOFs and its sensor head can pitch up and down but cannot yaw sideways. The arm kinematics and limited sensor head field of view played a role in shaping the solutions we devised to the manipulation problems faced in the DRC Trials. To obtain a rough idea of the robot's manipulation "sweet spot," we sampled 15^6 right-arm configurations, and rendered the end effector workspace using isocontours in Figure 2. Here red regions have the highest reachability score (~75 distinct sampled orientations) and blue the lowest (~5 or fewer distinct sampled orientations). In the left figure, the horizontal field of view of the right head camera is shown.

One significant practical effect of limited reachability and sensor coverage on task development was that most object manipulation occurred outside the view of the high-resolution head cameras. As a result, we were forced to rely heavily upon the low-resolution images produced by the situational awareness cameras and LIDAR. The relatively short length of the arms also made picking up objects on the ground while standing nearly impossible. We therefore opted to use 15 cm wrist extenders for the debris task, in which a longer reach was advantageous.

2.3. Kinematic Calibration

Precise and accurate positioning of the end effector was a critical capability for many of the Trials tasks involving manipulation. Combined with high-fidelity perception, accurate reaching reduced or eliminated the need for manual adjustment of the end effector by the operator. To minimize forward kinematic error, we performed detailed calibration of the upper body joints. The arm and neck joints are each outfitted with two position sensors: a LVDT mounted on the actuator and a digital encoder on the linkage. Since the LVDTs are noisier and subject to up to two degrees of backlash in the linkage, we chose to calibrate the encoders and use them for forward kinematic positioning.

The kinematic calibration procedure of the n upper body joints was straightforward: VICON markers were placed on the upper torso of the robot and M markers on each of the end effectors. Data were collected from N different representative poses of the upper body. In the spirit of Khalil & Dombre (2004), and assuming that the initial calibration of the encoders was roughly correct, we then solved the unconstrained minimization problem:

$$q^* = \underset{q_c \in \mathbb{R}^n}{\arg\min} \sum_{i=1}^{N} \|\phi(q_i + q_c) - x_i\|^2, \tag{1}$$

where $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^{6M}$ is the Cartesian position of the end effector markers, relative to the robot torso, as predicted by the forward kinematics, $q_i \in \mathbb{R}^n$ are the measured, pre-calibration joint angles, $x_i \in \mathbb{R}^{6M}$ are the VICON measured marker positions, and $q_c \in \mathbb{R}^n$ are the proposed encoder offsets.

The calibrated offset q^* was then added to all encoder measurements to provide more accurate joint angle estimation. This calibration procedure reduced forward kinematic error to approximately 1 cm on average (from more than 3 cm originally), although error was dependent on the specific robot posture.

Despite calibration, uncertainty remained in the absolute encoder positions after the robot was powered off and on. To account for this, the postcalibration positions of the

Figure 1. The platform consisted of the Atlas humanoid robot (left, photo credit: Boston Dynamics), which could be combined with a variety of hands or hooks via machined mounting plates. Right, clockwise: the iRobot and Robotiq hands, a hook for climbing, a 15 cm wrist extender, and a pointer for pushing buttons.

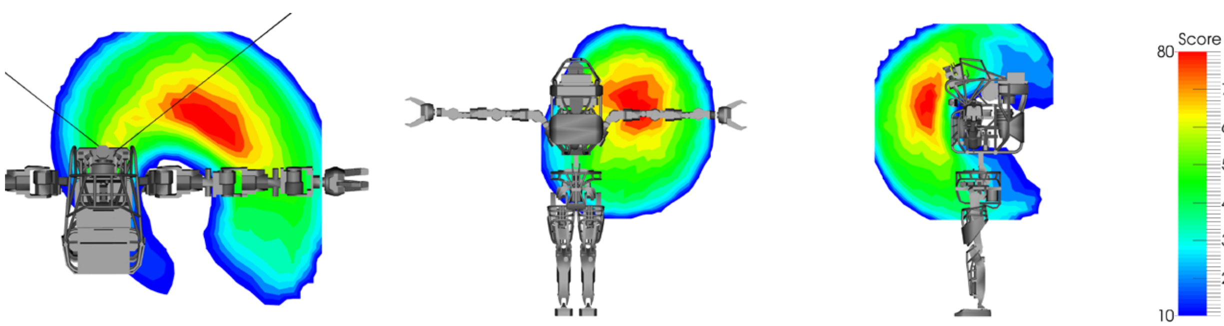


Figure 2. Visualization of the kinematic reachability of the robot from inferior (left), anterior (center), and medial (right) perspectives. We uniformly sampled 15 angles in the feasible range of each of the six right arm joints, divided the reachable workspace into $20 \times 20 \times 20$ bins, and assigned each bin a score based on how many distinct end effector orientations it contained.

encoders were measured and recorded at known joint limits. Upon robot startup, an automated routine moved the joints to these known positions and updated the calibration data.

2.4. Perception Calibration

Perception in the Trials primarily relied on (1) a pair of cameras in binocular stereo configuration mounted on the head; (2) a laser range scanner mounted on a rotating spindle on the head; and (3) a pair of wide field-of-view (FOV) cameras mounted on the chest. To place data from these sensors into a common reference frame for fusion and operator situational awareness, it was necessary to determine accurate geometric relationships among them.

The head stereo cameras and laser were factory-calibrated, although we modified the rigid laser-to-spindle and spindle-to-camera transformations slightly to achieve better scan-to-scan and scan-to-scene consistency. This was achieved by collecting stereo depth maps and laser scans of a calibration target while rotating the spindle through 360 degrees. Nonlinear least-squares optimization via the Levenberg-Marquardt algorithm determined the transformations that minimized laser point dispersion on the three mutually distinct surfaces of the target (Figure 3).

We determined each chest camera's intrinsic lens parameters (focal length, projection center, and fisheye distortion profile) by applying a variant of Hartley's parameter-free radial calibration method (Hartley & Kang, 2007) to images of a known planar pattern, which enabled very accurate rectification over the entire field of view as well as synthetic planar projections that were more easily interpreted by the operator (Figure 4). We determined each camera's extrinsic rigid pose with respect to the robot's torso by first aligning it with the head, then inferring the camera-to-torso transformation via forward kinematics. The algorithm discovered SIFT feature matches between the head

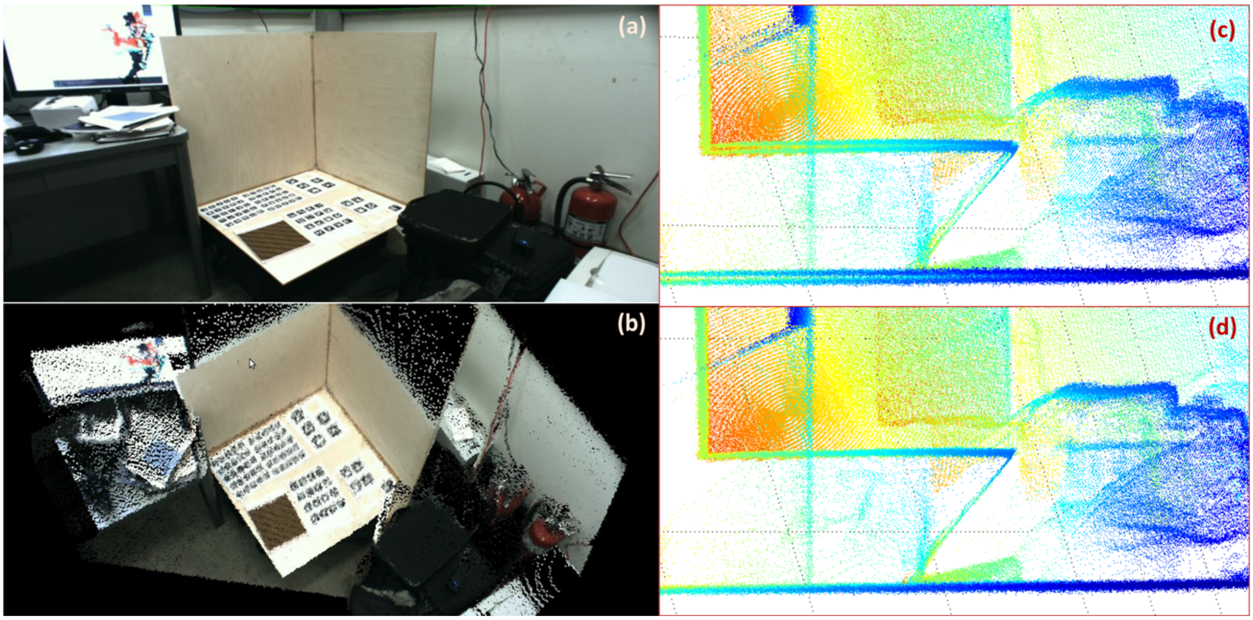


Figure 3. Head sensor calibration refinement. (a) Image of our calibration target acquired by the left head camera. (b) Postcalibration rendering of the left head camera image projected onto a LIDAR point cloud, demonstrating accurate alignment between laser and camera. (c) Side view of a 360-degree LIDAR point cloud sweep before spindle calibration; flat surfaces (ground, target planes) appear doubled due to slight spindle misalignment. (d) Coherence of flat surfaces improves after spindle calibration.

Figure 4. Fisheye lens rectification. (a) Raw image from the left chest camera exhibits severe fish-eye distortion and is difficult to interpret at the periphery. (b,c) After lens calibration, images can be rectified to produce arbitrary virtual viewpoints with minimal distortion.

and chest cameras' overlapping fields of view, and associated each matched feature with a laser return, thus producing a set of three-dimensional to to-dimensional (3D-to-2D) correspondences. A PnP (resection) solver wrapped in a RANSAC outer loop then used these correspondences to find a robust solution for the pose of each chest camera (Figure 5).

3. SOFTWARE SYSTEM OVERVIEW

The system we developed for the DRC Trials consisted mainly of software created during the simulation phase of the competition (Tedrake et al., 2014). Some components, such as the IK engine (Section 4.1) and height map software (Section 3.2.3), transferred to the physical robot without modification, while some task-specific planning interfaces, the model fitting interface (Section 5.1), and various controllers were developed after the Atlas robot arrived.

Our multiprocess software infrastructure involves modules communicating with one another in soft real-time using lightweight communication and marshalling (LCM) message passing (Huang et al., 2010). LCM is implemented using UDP and supports a variety of language bindings, pairing well with our code base, which was a heterogeneous mixture of C, C++, Python, Java, and MATLAB. Communication contracts between processes were instantiated as

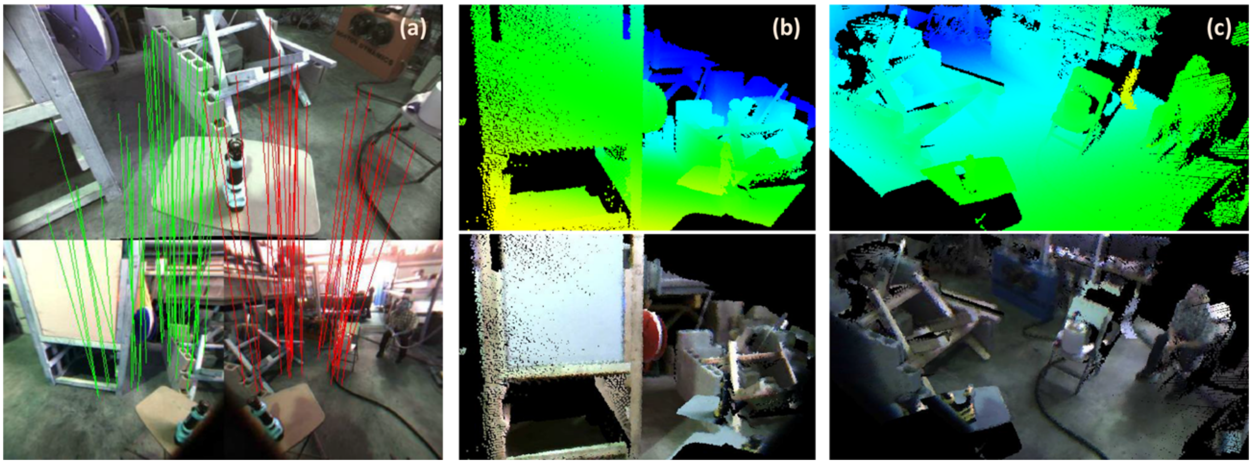


Figure 5. Chest camera alignment. (a) SIFT feature matches extracted between head image (top) and left/right chest camera images (bottom) used to align the sensor frames. (b,c) LIDAR points pseudo-colored by distance (top) and texture-mapped with chest camera images (bottom), demonstrating accurate alignment between head and chest data.

LCM message definitions, which allowed independent development of modules.

3.1. System Layout

Our system architecture is illustrated in Figure 6 with its network layout defined by DARPA's competition rules. The robot's onboard computation resources were reserved for Boston Dynamics' proprietary software, communicating over a 10 GB/s fiber-optic network with a series of field computers. This released teams from current computational constraints under the assumption that these field computers will eventually be transferred onto the robot in a future platform.

The human operators were separated physically from the robot (including the field computers) and had limited communication with it. Typically, two team members (one primary operator and one perception operator) interacted with the robot using standard Linux workstations that comprised the operator control unit (OCU). Additional machines could be easily connected to the LCM network as required to monitor performance or to run motion planners. LCM communication was unrestricted within the OCU and robot subnetworks; by contrast, data transmitted between the networks were carefully managed (Section 3.3).

One design feature of our system was that, because all message channels on the OCU and field computer were consistent, we could collapse the system back to a single machine by removing the network separation between the OCU and field computer. Therefore, during typical lab testing, developers would operate the robot solely from the field computer with high-frequency, low-latency communication, which allowed us to directly compare the envisaged and actual execution of plans.

As the competition approached, we limited ourselves to always using our network shaping software so as to transmit fewer data between the robot and the OCU and to add realistic latency, but by design operators observed no major difference except for having decreased resolution point clouds, reduced quality imagery, and lower-frequency robot state information.

3.2. Major Components

In this section, we describe the major components of our system and their interactions at a high level. Subsequent sections treat these components individually in detail.

3.2.1. Atlas Interface

Boston Dynamics provided each competing team with a minimal software interface to the core Atlas robot (colored blue in Figure 6), which included the ability to command the robot at a behavioral level (to dynamically walk, statically step, or balance) as well as at a low level (to control all 28 actuators).

Messages from the onboard robot computer arrived at 333 Hz over a fiber-optic ethernet at a typical bandwidth of 1.1 MB/s, providing data on the progress of these high-level commands as well as lower-level sensor and system readings. Upon receipt of these data, our system combined (green) the robot's pelvis pose and velocity estimate with joint sensor data from each component, incorporating calibrated upper body joint encoder data (Section 2.3) with hand and head joint angles. This represented our core *robot state*, which was a primary input to all other system components.

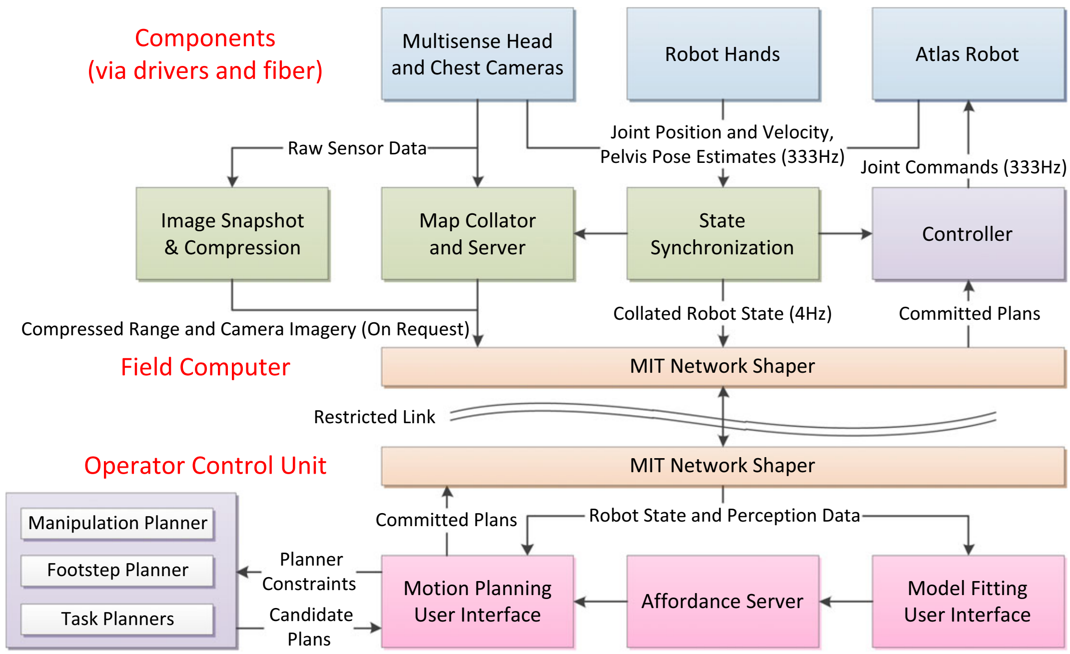


Figure 6. The major software components of our system. A series of processes running on the robot collated kinematic and perception sensor data, and responded to operator requests for LIDAR and imagery (green). The operators interpreted this data (pink) and formulated motion plans (purple), which were executed by the robot controller (purple). Data was transmitted across DARPA's restricted link by our network shaper software (orange).

3.2.2. Sensors

The stereo cameras on the sensor head (blue) provided uncompressed color images and disparity maps at 30 Hz at a resolution of 1,024 × 512 (or about 13 pixels per degree) at a maximum data rate of about 100 MB/s. Two situational awareness cameras, located on the upper torso, provided a hemispherical view of the scene at much lower resolution (1,280 × 1,024 or about 4 pixels per degree). By comparison, the spinning Hokuyo sensor provided 1,081 range returns per scan at 40 scans per second, for a total of 340 kB/s. Data streams from all devices were accessed via the fiber-optic ethernet, mildly compressed, and retransmitted internally to our robot-side LCM network. The total sustained data rate within the robot-side LCM network was about 5 MB/s from about 4,000 individual messages per second.

3.2.3. Perception Data Server

The perception data server (PDS) collected, processed, and disseminated raw sensor feeds from the robot's onboard head camera, chest cameras, and laser scanner. The server maintained a time-tagged history of recent data, including 3D LIDAR point clouds, minimally compressed color images, and stereo disparity maps, which could then be requested by the operator or by other system components for analysis and visualization. Broadly, two types of views—3D snapshots and images—were supported.

Downlink bandwidth limitations necessitated tradeoffs between data size and fidelity; the PDS was therefore designed to support a number of different representations and compression strategies that could be customized to suit particular tasks. Due to unknown network reliability—in particular, the possibility of dropped messages—we chose to adopt a stateless paradigm in which each unit of data sent from the robot to the operator was independent, so that loss of a particular message would be quickly ameliorated by subsequent successful transmission. Although this design choice precluded the use of temporal compression (e.g., H.264 video encoding) and sometimes resulted in transmission of redundant information, it proved to be quite robust and simple to implement, and it provided low-latency situational awareness to the operator. A summary of the most commonly requested products is provided in Table I.

The most recent laser range data were accumulated within a sliding temporal window of roughly 10 s (and typical used only when stationary due to state estimation drift). However, rather than sending raw laser returns or point clouds to the operator, the PDS consolidated a subset of scans into a single customized "view" that reduced redundancy, especially along the spindle rotation axis where point density was very high. View specification comprised a number of parameters, including spatiotemporal volume (in the form of time range and bounding half-planes), format (range image, height map, resampled point cloud, or

Table I. Compression strategies for the most frequently used data products. Typical message sizes from the competition (in bytes) are reported on the right.

Product	Resolution (pixels)	Compression	Size (bytes)
low-quality head image	256 × 136 (4× downsample)	jpeg 50% qual	4,900
high-quality head image	512 × 272 (2× downsample)	jpeg 50% qual	16,200
low-quality chest image	256 × 320 (4× downsample)	jpeg 50% qual	7,600
foveated image chip	<200 × 200 (no downsample)	jpeg 70% qual	4,700
workspace range map	200 × 200 (1–2 cm spacing)	8-bit quant, zlib	8,300
terrain height map	200 × 133 (3 cm spacing)	8-bit quant, zlib	8,600

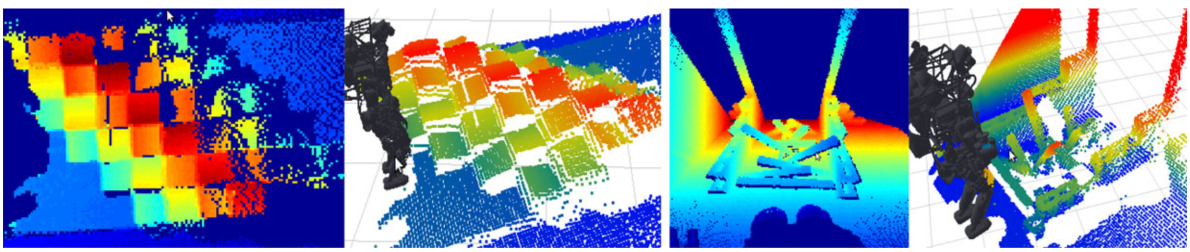


Figure 7. Sample composite views of raw laser range data. From left to right: a height map of uneven terrain acquired during the Walking task, and its decoded 3D point cloud representation; a range map and decoded point cloud of the workspace in front of the robot acquired during the Debris task.

octree), data source (raw LIDAR, filtered LIDAR, or stereo disparity), viewpoint, and resolution. Each view also carried an associated spatial transformation describing the geometric relationship between its data and the world coordinate system. Once formed, the view was encoded for transmission at the specified message frequency via tunable spatial quantization and lossless zlib compression. Figure 7 depicts typical height and range maps produced by the server.

Raw color images from the head and chest cameras were also cached and transmitted upon request. The PDS could adjust spatial downsampling, JPEG image quality, and message frequency to accommodate link constraints; it could also send an accompanying high-resolution, low-compression image chip covering a small operator-specified region of interest, which was reassembled at the operator station to form a foveated image without consuming substantial additional bandwidth (Figure 13).

3.3. Networking

Communication between the OCU and the robot was highly restricted in the DRC Trials in order to simulate the effects of a real network in a disaster zone. To add some time variance to the channel, two bands were created by the contest designers representing a relatively high quality link (1 Mbps throughput and 50 ms one-way latency) and a more degraded link (100 Kbps throughput and 500 ms one-way latency). These bands were used in the competition in alternation, with DARPA swapping the band in use for the other band once per minute.

To address the throughput restrictions, we developed a tool called the *network shaper* that used source encoding, decimation, and priority queuing of messages destined to travel between the OCU and the robot. This tool relied on the Goby2 communications library, which was originally developed for very low-rate underwater acoustic data links (where average throughput is on the order of 100 bits per second), and it was thus suitable for extension to this somewhat higher data-rate regime. Goby2 includes the Dynamic Compact Control Language (DCCL), which is an interface description language (IDL) coupled with a set of extensible source encoders for creating very small messages by taking into account the precise bounds of the data values to be transmitted in a given message's fields. The modest amount of added latency was ameliorated by the operator developing each plan locally and transmitting it to the robot, which executed the plan autonomously as described in Section 4.

Throughput and latency limits were implemented using a separate computer (the "Mini-Maxwell" developed by InterWorking Labs) with a network bridge running Linux and the network emulator ("netem") traffic queuing discipline ("qdisc"), which can be manipulated (in testing) with a traffic control ("tc") tool. The particular implementation used for the competition incorporated large buffers that held 10 s of data after the throughput limit was reached. Since this buffer length was one to two orders of magnitude

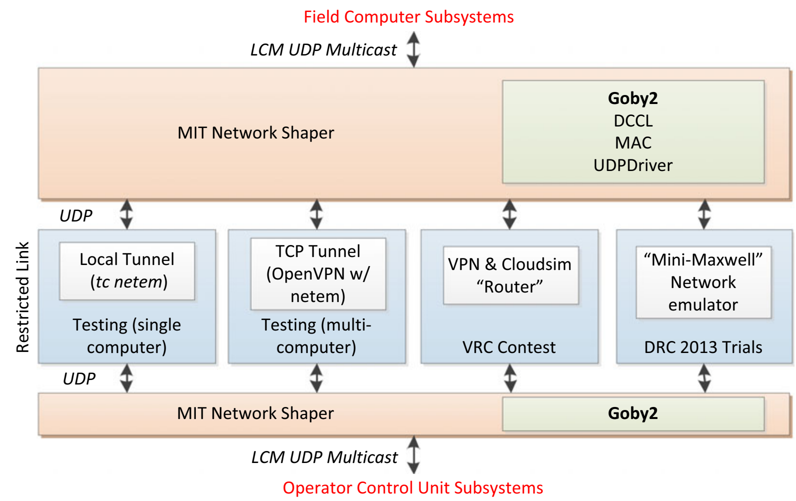


Figure 8. Detailed view of the networking components of the MIT DRC software system (full system in Fig. 6). To simulate network degradations that might occur between operators and the robot in a disaster zone, various tools were used to artificially restrict throughput and increase latency. The IP tunnels shown on the left were developed by our team to perform full robot simulations before the contest tools were completed. To transfer data efficiently between the OCU and the robot, compression (source encoding) and queuing were performed by the network shaper. Packets were transmitted using UDP to avoid the overhead of TCP.

greater than the latency introduced by the undercapacity link, timely operator response required ensuring that the throughput limit was never reached. Since the buffers were so large, it was not feasible to automatically detect available throughput by measuring packet loss, as the discovery time would be on the order of the network band alternation time. Thus, the network shaper autonomously detected the current band in use by measuring the latency of every packet and rate-limited all outgoing packets to avoid exceeding the Mini-Maxwell's buffers.

This approach was effective until the latency/bandwidth settings were accidentally switched by the contest operators, such that the 1 Mbps throughput regime was paired with a 500 ms latency setting and the 100 Kbps throughput was paired with a 50 ms latency. During the 100 Kbps band, data would fill the buffers of the mini-Maxwell, leading to 10-s latencies at the OCU. To compensate during these periods (we were not aware of this inadvertent switch until the run began), the operator was able to use the data request functionality in our interface to manually throttle downlink bandwidth and avoid saturating the buffers. This experience stresses the importance of properly designing the underlying network link between the OCU and the robot, and it has led us to believe that small buffers would be preferable for avoiding artificially high latencies and for rapid automatic detection of network throughput. While the settings of the mini-Maxwell were outside the control of our team for the competition, a real disaster-zone network could easily be designed to have modest buffers because, unlike physical effects such as latency and packet loss, buffering is an intentionally designed aspect of a network.

Figure 8 shows a diagram of the system components related to networking. For testing purposes, we developed our own restricted network tool (also based on "netem" over a layer 3 IP tunnel) that could allow single or multiple machine simulations of the entire operator/robot system.

3.4. System Management

In the VRC we utilized two robot-side computers (focused on control and perception, respectively) and up to three base-side UIs, while in the DRC Trials our control and perception subsystems ran on a single robot-side computer.

To manage this distributed computing architecture, we utilized another tool developed during the DUC called *procman* (Leonard et al., 2008). Via a simple user interface, called *procman sheriff*, operators could launch our system on each of the host machines (called *deputies*) with a single mouse click, monitor memory usage and process behavior, and manage computation load. Procman also allowed us to easily keep track of the robot model configurations we used in different tasks.

Because of the restricted communication channel, we modified procman to ensure that no recurring bandwidth usage occurred: a summary of the system status was delivered only upon a detected change, rather than continuously, and process names were hashed rather than being sent in plain text.

4. MOTION PLANNING

Our planning infrastructure enabled us to pose and solve a wide variety of motion planning problems. The two key components of our approach were an efficient, optimization-based inverse kinematics (IK) engine and an automated footstep planner. The former allowed us to describe the requirements for a given motion in terms of a library of kinematic constraints. Our IK engine translated these constraints into a nonlinear program that was then solved to yield a feasible motion plan. The footstep planner converted user-specified navigation goals into a sequence of footsteps that respected constraints on the relative placement of successive footsteps. This section describes the capabilities of these planners, which provided the language into which our user interface translated the user's intent.

4.1. Efficient Inverse Kinematics Engine

Our specification for a kinematic motion planner required that it be efficient, i.e., capable of producing feasible whole-body plans for Atlas in much less than a second, and expressive, i.e., capable of reasoning about a variety of constraints and goals. Although efficient Jacobian-based algorithms for computing local IK solutions exist [e.g., Buss (2004)], they cannot easily incorporate arbitrary objectives and constraints. Therefore, we cast IK as a nonlinear optimization problem that we solved locally using an efficient implementation of sequential quadratic programming (SQP). Results generated by the IK engine were of one of two types: 1) a single robot configuration, or 2) a smooth time-indexed trajectory of configurations.

Single-shot IK: For a single configuration problem, the objective was to minimize the weighted L_2 distance from a nominal configuration,

$$\arg\min_{q\in\mathbb{R}^n}(q_{\text{nom}} - q)^T W(q_{\text{nom}} - q),$$

subject to

$$f_i(q) \le b_i, \quad i = 1, \dots, m,$$

where $q_{\text{nom}} \in \mathbb{R}^n$ is a nominal configuration of the robot in generalized coordinates (for Atlas, $n = 34$), and $f_i(q) : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \dots, m$, is a set of kinematic constraint functions. This type of optimization can be used to, e.g., find a configuration for reaching some target position while remaining as close as possible to the current standing posture.

Trajectory optimization: For many problems of interest, finding a single configuration is not sufficient. Rather, we would like the robot to achieve a complete motion that is smooth and respects some set of (possibly time-varying) constraints. To generate configuration trajectories in this way, we formulated an optimization problem over a finite set of configurations, or *knot points*, interpolated with a cubic spline. The trajectory optimization problem had the form

$$\arg\min_{q_1,\dots,q_k}\sum_{j=1}^{k}(q_{\text{nom},j} - q_j)^T W(q_{\text{nom},j} - q_j) + \dot{q}_j^T W_v \dot{q}_j + \ddot{q}_j^T W_a \ddot{q}_j,$$

subject to

$$f_i(q_1, \dots, q_k) \le b_i, \quad i = 1, \dots, m,$$

where additional costs associated with the generalized accelerations and velocities at each knot point were added to promote smooth trajectories.

The API to the IK solver allowed the user to specify a variety of constraints to flexibly generate whole-body configurations and trajectories. The supported constraint types included:

- joint limits
- pose of end effector in world/robot frame (bilateral and unilateral)
- cone (gaze) constraints of end effector in world/robot frame
- distance between two coordinate frames
- center-of-mass position and quasistatic constraints
- linear configuration equalities (e.g., left/right symmetry)

Here "end effector" should be interpreted to be any frame attached to a rigid body. For trajectory optimization, it was also possible to specify time windows during which subsets of constraints were active.

IK problems were solved using an efficient SQP implementation that exploits the structure shared by most problem instances. Due to the tree-structure of the robot's kinematics, the constraint gradients are often sparse. For example, a constraint function on the foot pose is unaffected by changes in the position of the torso DOFs. Using SNOPT (Gill, Murray, & Saunders, 2002), we were able to solve single-shot IK problems in approximately 10 ms and find smooth motion trajectories with five knot points in 0.1–0.2s. As discussed in Section 7.1, this was essential to enable online planning of entire motions.

During the competition, we took advantage of the presence of the user by not automatically detecting collisions or planning around obstacles. Also, on occasion, the solver was not able to find feasible plans, for example when commanded to reach unreachable locations. When a plan that would result in collision or was unable to reach its goal was proposed, the user was then required to make adjustments to the constraints or to plan around the obstacle and resubmit the request to the planner. (In Section 8.3, we discuss our future plans to incorporate obstacle avoidance within our planning optimization.)

Optionally, the planner could successively relax constraints, such as the end effector pose, until a feasible plan was found. In this case, the feasible plan would be returned

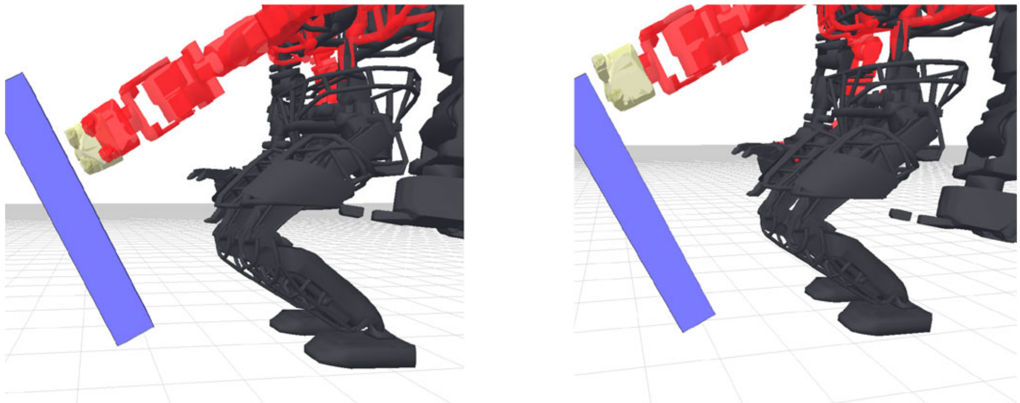


Figure 9. Relaxation of reaching plans to the affordance (blue) by the robot (gray). Left: a pregrasp reaching goal (yellow) just reachable (red) after automatic relaxation in position and orientation (by 2 cm and 3 degrees, respectively, in this case). Right: if the operator was unhappy with the relaxation, the reaching goal could be manually repositioned until precisely reachable. Here the planner is simultaneously solving for the six joints of the arm and the three back joints.

to the user along with information about the degree of relaxation. An example of a relaxed plan is illustrated in Figure 9. It was common for our operators to relax reaching plans in the debris moving task. With many of the pieces placed below the robot's knees, there where many cases in which an ideal placement of the robot's hand flat against a debris piece was not possible, forcing the execution of a suboptimal grasp.

4.2. Planner Classes

To promote faster and more predictable interaction between the operator and the IK engine, we developed several planning classes that employed canonical sets of constraints and objective functions. This allowed the operator to succinctly specify inputs to the planner through the user interface (Section 5) rather than having to completely construct planning problems before each action.

The *manipulation planner* responded to end effector goals generated by the operator. The input could be of the form "reach to this hand pose," which was converted to a position and orientation constraint on the active end effector at the final time. In this case, the manipulation planner returned a time-indexed joint-angle trajectory from the robot's current posture to one that satisfied those constraints. The operator could specify the behavior of the rest of the body during the reach by selecting from two planner modes: one that kept all other end effector poses fixed, and one that allowed whole-body motion.

It was also possible to input a trajectory of end effector goals to generate a manipulation plan. For example, as illustrated in Section 5.2.2, the operator could request a valve turning plan by simply rotating a valve affordance with a grasped hand attached, hence defining a desired end effector trajectory.

The *end-pose planner* responded to operator requests for a feasible configuration from which to perform a manipulation. Given a time-indexed set of end effector goals, the end-pose planner used the IK engine to simultaneously search for a smooth manipulation trajectory and a feasible constant standing posture from which the manipulation could be executed. This posture could then be used in turn to generate appropriate walking goals so that desired manipulation could be carried out after the robot finished walking to the object of interest.

The *whole-body planner* responded to inputs containing desired footstep locations, a set of end effectors to keep fixed, and high-level motion parameters such as desired foot clearance over the terrain map or bounds on pelvis sway. This planner generated smooth, whole-body configuration trajectories that satisfied the foot and end effector pose constraints in addition to quasistatic constraints that kept the robot's center-of-mass ground projection within the support polygon of the feet. This planner was used in the DRC Trials to climb the ladder.

Finally, the *posture planner* was the simplest planner. Its input was a desired joint configuration for the robot, for which it would return a smooth trajectory from the current configuration to the goal. This planner was useful for moving to a prescripted joint configuration that, e.g., retracted the right arm close to the body to enable subsequent movement through a tight space.

4.3. Footstep Planning

Somewhat separate from the IK-based planners, the footstep planner was used to determine automatic footstep placement given a high-level goal from the human operator, as well as detailed input from the user about the specific location of a particular footstep if required. Specifically, given

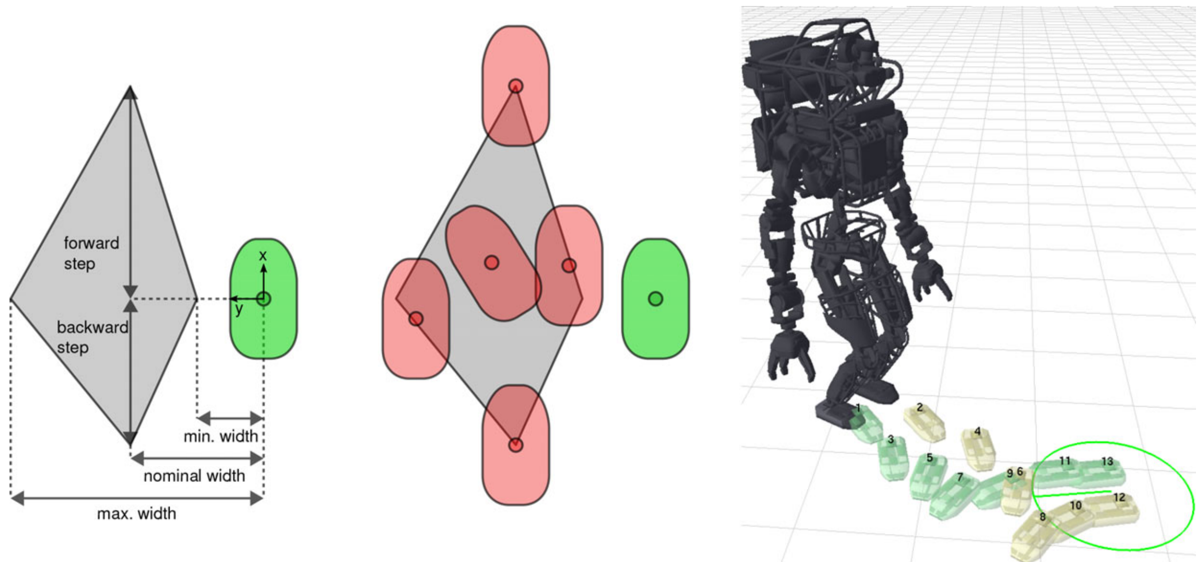


Figure 10. Left: A simplified representation of the kinematic reachability region for footstep placement. Given a fixed position and orientation of the right foot, we compute a polyhedral region defined by parameters *min width*, *max width*, etc., into which the next position of the left foot must be contained. In practice, this polyhedron is represented in four dimensions: x, y, z, and yaw, but only x and y are drawn here. Center: several example locations for the left foot (red) relative to the right foot (green). Right: A complete footstep plan from the robot's position to the target position and orientation showed by the green circular marker. In this particular plan, the robot steps forward, then to the side, then backward in order to reach the goal.

the robot's current configuration and a desired goal pose expressed as a position and orientation on the surface of the perceived terrain, the footstep planner computed a sequence of kinematically reachable and safe step locations to bring the robot's final standing position as close as possible to the goal, while allowing easy operator adjustment of the plan. The planner minimized the total number of steps required to reach the goal by exhaustively searching over all numbers of steps between a user-specified minimum and maximum, typically one and ten, respectively, and choosing the plan with the fewest steps that still reached the goal (or the plan that came the closest to the goal if none could reach it). Each search for a plan with a fixed number of steps required only about 10 ms, so exhaustive enumeration over a variety of numbers of steps was not prohibitive.

Kinematic reachability of a footstep plan was ensured by constraining the location of each step to within a polyhedron defined relative to the pose of the prior step. This reachable polyhedron was computed offline by fixing the position of one foot and running inverse kinematics on a wide range of possible positions for the other foot. The step locations for which the inverse kinematics problem were feasible were then converted to a polyhedral inner approximation, which provided a conservative, convex set of locations for the placement of one foot relative to the prior step location. A simplified illustration of this polyhedral approximation is shown in Figure 10. The primary planning algorithm used during the DRC was a simultaneous nonlinear optimization of the position and orientations of several steps, typically one to ten. The optimization attempted to minimize the error between the final steps and the goal, while respecting the polyhedral reachable regions. This approach differs from the footstep planning methods used by Huang (Huang, Kim, & Atkeson, 2013), Chestnutt (Chestnutt, Kuffner, Nishiwaki, & Kagami, 2003), and Kuffner (Kuffner, Nishiwaki, Kagami, Inaba, & Inoue, 2001) in that it performs a continuous optimization over the positions of the footsteps, rather than reducing the choice of step location to a discrete selection among a few precomputed relative step positions.

Once the planner had generated a candidate footstep plan, it was presented to the operator as a sequence of footsteps, as shown in Figure 10. The x and y positions and yaw of each step could be individually adjusted through the user interface, and the planner ensured that the roll, pitch, and height of each step matched the estimated terrain underfoot. The height map representation used in planning performed a RANSAC plane fit to fill in the gaps in the as-sensed height map. Since the LIDAR provided no data about the ground near or under the robot's feet, the height map used the forward kinematics of the robot's legs to find the height and orientation of the bottoms of the feet to estimate the terrain directly underfoot and within a 1 m square region around the robot.

During the VRC, we used an edge detection filter to classify areas of the terrain that were too steep or rough

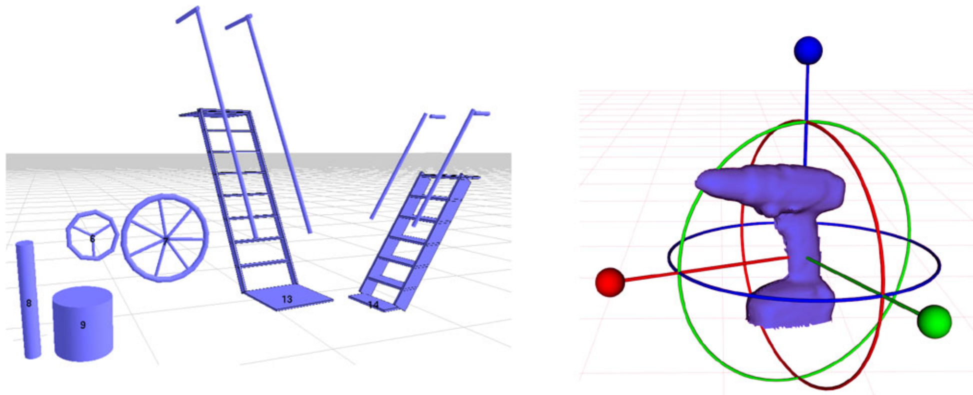


Figure 11. Affordances were represented in a parametrized XML format. Left: a collection of objects under three templates: (i) cylinder; (ii) valve; and (iii) angled ladder. Two pairs of objects are shown under each affordance class, which are instantiations with different parameters. Right: Interactive markers allowed the user to adjust the position of an affordance, or to instruct the robot to move the affordance along one of its internal degree of freedom.

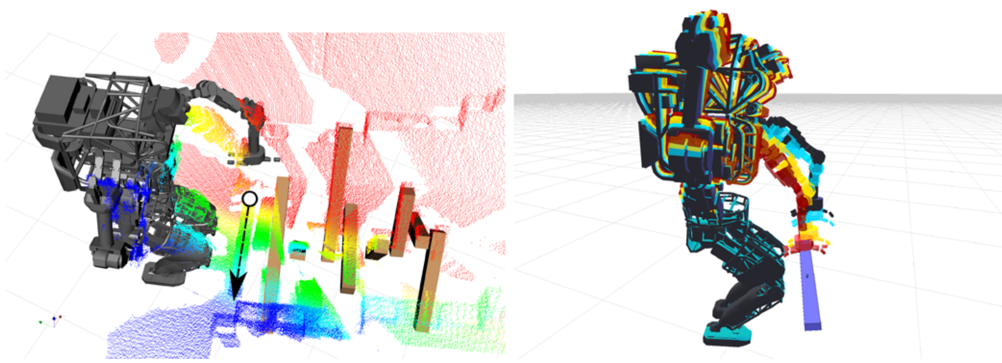


Figure 12. A fitting algorithm produces an affordance that can be used to plan a reach motion. Left: A line annotation (arrow) defines a search region that is used by a segmentation algorithm to fit a 2 in. × 4 in. board affordance to the 3D point cloud data. Right: An end effector goal placed on the board affordance is used to compute a reaching plan.

to step on, and the footstep planner was constrained to avoid placing steps on these regions. However, the generally flat terrain of the DRC made this approach less useful, and the extremely precise footstep positioning during the rough terrain task was primarily done manually, so this capability was not used during the DRC Trials.

5. USER INTERFACE

The rich motion planning system discussed in the previous section enabled us to design a robot interaction system at a higher level than traditional low-level teleoperation. Specifically, we sought to interact with the world at the level of *affordances*. We defined affordances as objects of interest that could be physically acted upon by the robot to perform some task-specific goal. These objects were specified as a series of links and joints organized in a tree structure similar to the well-known Unified Robotic Description Format (URDF) used to represent the kinematic structure of robots (Sucan, 2014). We extended this format to enable templatization and termed it the *Object Template Description Format* (OTDF). Figure 11 shows some examples of affordances that were described by the OTDF.

The primary robot operator could manually specify the parameters of these objects, or it could instead draw upon the output of a semiautomated LIDAR fitting system (controlled by the perception operator) that detected and positioned objects relative to the robot in our 3D interface.

5.1. Affordance Fitting

We developed both automatic and human-assisted object fitting algorithms for the competition (Figure 12). While we had some success with fully automated fitting, we focused more on human-assisted fitting due to the competition rules. In the DRC Trials, unlike the VRC, operator input (OCU-to-robot bandwidth usage) was not penalized, and the extra time required to provide human input to a fitting algorithm

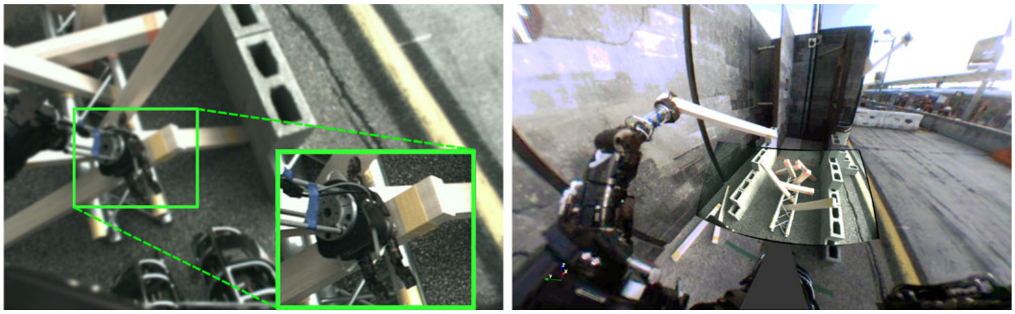


Figure 13. Left: an example of foveation during the debris task to enhance the operator's situational awareness without drastically impacting bandwidth. The operator selects a region of interest on a low-resolution image to request a high-resolution subimage. Right: a first-person combined view stitches the head camera image with the left and right fisheye chest camera images. The operator can use the mouse to zoom and look in all directions for superior situational awareness.

was small compared with the time lost due to planning robot motion with a poorly fit affordance model. Thus, human-assisted fitting algorithms were used in all five of the DRC Trials manipulation tasks.

The operator provided input by specifying initial search regions, and by accepting or adjusting pose estimation post-fit. Search regions were input either by 3D point selection in the LIDAR point cloud or by 2D annotations on point cloud projections or renderings. Images from all onboard cameras were fused and could be viewed interactively for visual context (Figure 13).

One human-assisted algorithm was developed for the debris task to enable rapid fitting of lumber of arbitrary dimensions laying in unstructured configurations with overlap and occlusions. The algorithm was designed to fit rectangular prisms, and it could be applied to boards, cinder blocks, bricks, and boxes. The operator began by selecting the expected cross-sectional dimensions of the affordance; for instance, lumber cross sections are typically 2×4, 4×4, or 2×6 in.

The operator then defined the search region by drawing a line annotation on the display using mouse clicks to define the end points. This annotation could be drawn either on a camera image or a 3D point cloud rendering; in practice, we always annotated using the 3D rendering because the camera's limited field of view did not often include the region of interest. The line end points were projected as rays in 3D originating from the camera center to define a bounded plane, and points in the LIDAR point cloud within a threshold distance of this plane were kept as the initial search region.

Given the initial search region, the board-fitting algorithm used RANSAC with a plane model to select inlier points that were then labeled as board face points. These were binned along the vector parallel to the edge connecting the search region ray end points, which was roughly parallel to the board's length axis. The algorithm identified one candidate edge point in each bin by selecting the maximum distance among that bin's points along the vector perpendicular to the RANSAC plane normal and length axis. Due to occlusions and adjacent surfaces, not all candidate edge points were true board edge points, so a RANSAC line fit was applied to the candidate edge points to label inliers presumed to be the board edge. The board affordance model was snapped to the fitted face plane and edge line, and extended along the edge line to encompass the minimum and maximum face point projections onto that line.

The resulting accurately fit object position, orientation, and scale parameters were finally transmitted to an *affordance server* that maintained the states of all identified affordances in the vicinity of the robot and made them available to both the user interface and the planners. Combined with the current robot state, this information provided inputs necessary for the operator to plan action sequences.

5.2. Primary User Interface

Figure 14 shows a screen capture of our primary user interface. The basis of the user interface was originally developed by MIT's DARPA Urban Challenge Team, and it has been reused in a number of projects internally. For the DRC, we have expanded this interface by significantly extending the input modes afforded to the user.

Within the main viewing pane, the operator perceived the robot's vicinity, queried planners, reviewed plans, and monitored plan execution. A robot avatar, the current affordances, and views of the LIDAR data were also rendered in 3D in the main viewing pane. From the side panels, the user could interact with plans, request new sensor data, and change data visualization properties.

5.2.1. Interaction with Affordances

The operator interacted with affordances primarily to specify inputs to the task planners. A double-click would launch a user menu that was unique to each object instance. The

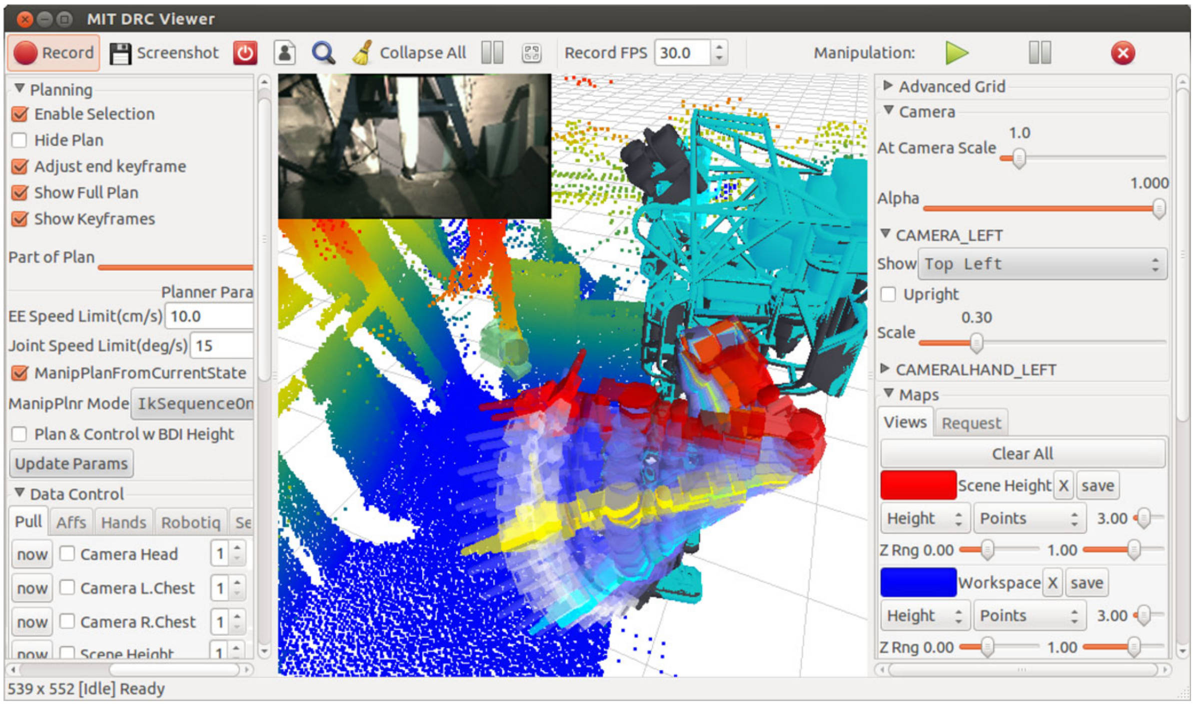


Figure 14. The primary user interface showing the 3D robot avatar while the operator planned a pregrasp reach to a hose. The candidate motion plan contains modifiable *key frames* in red, yellow, and teal (Section 5.2.2). A grasp goal or *sticky hand* (green) is fixed to an (obscured) model of the hose. Also visible are controls for executing the plan, making perception data requests immediately or at a specified frequency, and adjusting image quality and workspace viewport. 3D data views could be toggled, resized, or color-mapped (in this case by height).

menu enabled the operator to perform tasks such as adjusting the pose of the object; adjusting the internal state of the object (e.g., joint DOF positions); seeding grasps, footsteps, or gaze goals; searching for robot postures to achieve these goals; and finally loading corresponding goal seeds from storage. Interactive markers (shown in Figure 11 for a drill) could be used to adjust the pose of the object and its internal state. We also implemented a 3D mouse interface for some of these tasks, but our experiments with it did not demonstrate a significant advantage over the combination of a traditional mouse and keyboard.

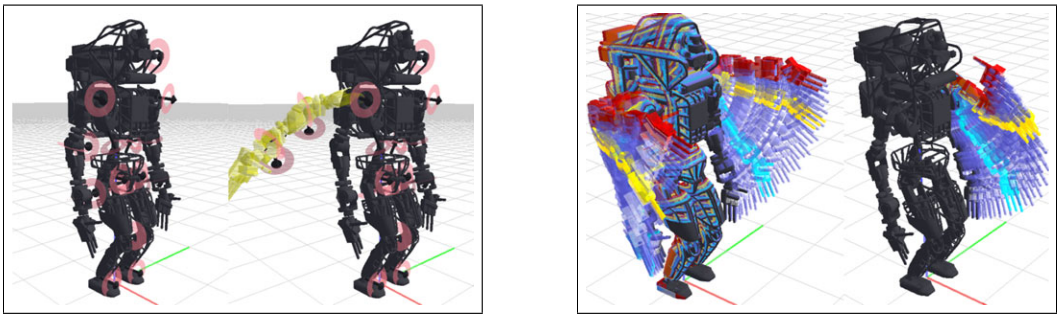
User intent was specified through phantom end effectors positioned relative to an affordance (Figures 14 and 15). In the case of grasping, the user could query an optimizer by selecting the position on the object where s/he wished the robot to grasp. The optimizer solved for a feasible grasp that satisfied force closure constraints. Alternatively, the user could take advantage of prespecified canonical grasps for known objects; for example, after fitting a door frame, the operator would immediately be presented with the location of a pregrasp hand position relative to the door handle.

These phantom end effector seeds, termed *sticky-hands* and *sticky-feet*, provided a mechanism to transform affordance-frame commands into end effector commands, which are more easily handled by the robot-centric planning and control modules. The seeds could be committed to planners as equality constraints (or desired goals) or as desired posture set points for grasp control.

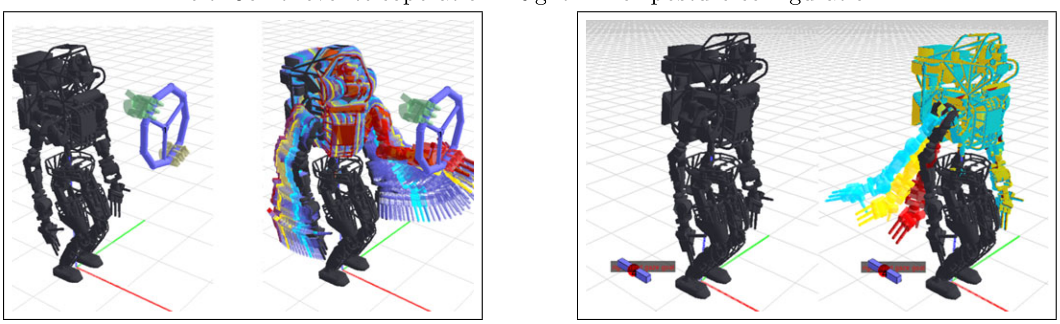
5.2.2. Interactive Planning

The user interface permitted bidirectional interaction between the operator and planners. An important consequence of this interaction was that input from the operator could be used to guide the planner away from undesirable local solutions by refining a proposed plan with additional constraints. A key feature that enabled this bidirectional interaction was *key frame adjustment*, which provided a mechanism for performing replanning interactively.

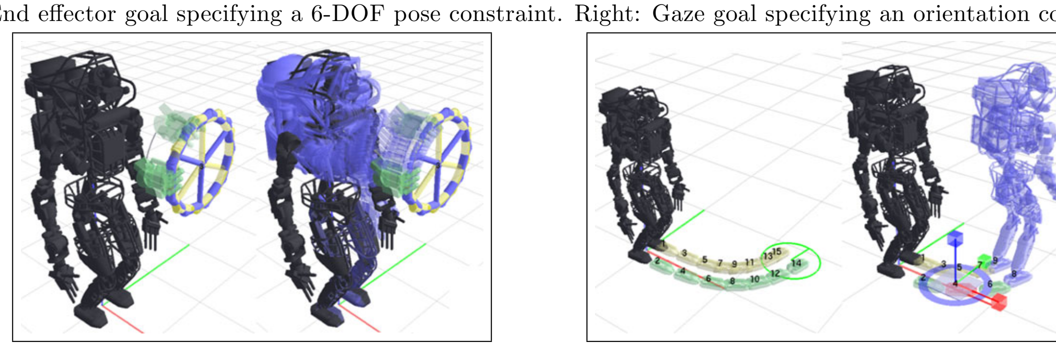
Key frames were samples from a proposed robot plan with which the operator could interact. Examples are illustrated in red, yellow, and teal in Figure 14. By modifying the location of an end effector in a key frame, the user could add position constraints to the planning problem. This allowed the operator to guide the optimization algorithm to plan collision-free motions around workspace obstacles.



Left: Joint-level teleoperation. Right: Prior posture configuration.



Left: End effector goal specifying a 6-DOF pose constraint. Right: Gaze goal specifying an orientation constraint.



Left: Sequence of poses to generate motion path. Right: Navigation goal specifying a destination pose.

Figure 15. Our interface supported a variety of user-specified goals. Here, a number of modes are illustrated with increasing complexity.

5.2.3. Additional Components

The UI also provided simple controls to manage and visualize data provided by the perception data server described in Section 3.2.3. Figure 14 depicts relevant UI panes responsible for server data selection (i.e., which particular camera images and 3D data view presets to request). The user could also customize the display of 3D data received from the robot, controlling visual representation (e.g., point cloud, wire-frame, or mesh) and 3D color coding (e.g., texture map from the cameras or pseudocolor by height or by range); we typically used point clouds and wire-frames with height-based color mapping. To Control of sensor settings (including camera frame rate, gain control, and LIDAR spin rate) and perception compression macros (“low”, “medium”, or “high” quality) was also available.

A secondary graphical user interface was used primarily by a second operator for object refitting and presenting alternative views, such as the stitched camera view shown in Figure 13. Finally, a simple keyboard interface was used to make minor but precise adjustments to the end effector position and orientation in several tasks.

6. PLAN EXECUTION

Operator-approved plans were carried out by streaming references from the field computer to controllers executing on the robot. By combining low-level position control with feedback from the arm joint encoders, we achieved single centimeter end effector position error. For each joint, the hydraulic valve input current, $c \in [-10, 10]$ mA, was computed by a simple PID rule:

$$c = K_p(q_{\text{ref}} - q) + K_d(\dot{q}_{\text{des}} - \dot{q}), \tag{2}$$

$$q_{\text{ref}} = q_{\text{des}} + K_i \int_0^t (q_{\text{des}} - q_{\text{enc}}) d\tau, \tag{3}$$

where q and $\dot{q}$ are the filtered joint position and velocity signals computed from the actuator LVDTs, and q_{enc} is the joint encoder measurement. Due to API restrictions, the robot servo loop signals were limited to LVDT-based joint positions and velocity estimates, so the PD rule (2) was executed onboard the robot at 1,000 Hz, while the references (3) were computed at 333 Hz and streamed to the robot from the field computer. Due to significant velocity noise in the unfiltered $\dot{q}$ signal (some arm joints had noise standard deviations greater than 3 rad/s), we used aggressive 5 Hz first-order low-pass filters to prevent undesirable high-frequency effects in the control loop. The time-varying desired joint positions and velocities, q_{des} and $\dot{q}_{\text{des}}$, were evaluated directly from the smooth joint trajectories generated by the planner.

We exploited the robot's available balancing and stepping behaviors to produce manipulation and walking motions. Coordinated upper-body and pelvis motions were achieved by streaming pelvis pose references to the balancing controller. Likewise, footstep plans were used to generate appropriate references for the stepping controller. For ladder climbing, position control of the entire body was used to follow whole-body, quasistatic climbing trajectories. A primary focus of our DRC Finals effort is replacing these behaviors with a custom whole-body, dynamic controller for manipulation and locomotion that will increase execution speed, stability, and reactivity. More details about this control approach are described in Section 8.3.

7. DRC TRIALS

We fielded our system in the DRC Trials held at Homestead Speedway in Miami, Florida on December 20–21, 2013. They represented the largest ever demonstration of robot multitask capability in an outdoor environment and were far removed from laboratory conditions. The Trials served as an intermediate progress evaluation and funding down-select before the DRC Finals expected to take place 12–18 months later. We competed against six other teams using a Boston Dynamics Atlas, and nine teams who built or brought their own robots. We finished in fourth place with 16 points—behind Schaft of Japan (27 points), the Florida Institute for Human and Machine Cognition (IHMC) (20 points), and CMU's CHIMP robot (18 points). IHMC used an Atlas robot.

The following table summarizes the outcome of the eight tasks in the order in which we attempted them. Teams had 30 min to complete each task. One point was scored for completion of each of three ordered subtasks; completing all three subtasks without intervention earned a bonus point. Interventions could be called by the robot operator or by a member of the field team, triggering a minimum 5-min penalty period during which teams could reset their robots behind a checkpoint. We attempted all eight tasks and scored in all but one (driving), achieving the maximum four points in the drilling and valve tasks.

To give the reader a better understanding of the operation of the robot through our system, we provide a detailed account of our execution of the drill and terrain tasks at the Trials in the next section. To summarize the remainder of our Trials experience, we highlight a few noteworthy observations and issues that occurred:

- In the debris task, a mistake while dropping two debris pieces required our operators to adjust their high-level task sequence and pick up a piece of wood at the boundary of kinematic reachability and sensor visibility. This effort allowed us to score a point in the last remaining seconds, and was a strong, albeit stressful, demonstration of our system's ability to plan and execute motions in unanticipated situations.
- In the hose and valve tasks, a misconfiguration in the competition network resulted in long communications blackouts. For example, after confirming a footstep plan to approach the hose, we did not receive a sensor update from the robot until it was standing in front of the hose. We were pleased that our system and operators could deal gracefully with this unexpected situation, with the sole side effect being slower task execution.
- As with other teams, wind was an issue for us. In the door task, our operator innovated a two-handed strategy to push open the first door, constantly applying pressure against it to prevent it from blowing closed. In the terrain and climbing tasks, we repeatedly had issues with the wind interfering with robot IMU initialization.
- As illustrated in Figure 16, lighting conditions were very challenging. Strong shadows made even human-aided perception difficult. We had anticipated this and therefore relied heavily on LIDAR-based perception with no automated vision algorithms. Nonetheless, with low-rate (~1 Hz) LIDAR updates, this approach has a limited capacity to support fluid, closed-loop control and perception.
- In the climbing task, we used steel hooks instead of actuated hands to brace the robot's considerable mass while ascending the ladder. Similarly, using simple unactuated metal pointers offered the most robust solution for opening doors, pressing buttons, and turning valves. The fact that nearly every Atlas team adopted similar appendages for these tasks suggests that dexterous hands capable of greater strength and stiffness might significantly improve the versatility of disaster robots.

Table II. Wall drilling task.

Task	Score	Objective	Summary
Debris	1 pt	Clear 10 pieces of wood, walk through corridor	Cleared five pieces of wood, recovered from dropping two near robot's feet
Hose	2 pts	Grasp and mate hose with a standpipe	Grasped hose, failed to mate hose with standpipe
Valves	4 pts	Turn one lever valve and two wheel valves	Succeeded without intervention, despite erroneous network behavior
Drill	4 pts	Grasp and turn on a drill, cut triangle pattern in wall	Succeeded without intervention, see Section 7.1
Door	1 pt	Open and walk through three doors	Opened and walked through one door, fell over unexpectedly
Vehicle	0 pts	Drive a utility vehicle through course, exit vehicle	Drove approximately half the course before time expired
Terrain	3 pts	Walk across increasingly difficult terrain	Succeeded after an intervention, one of three teams to finish
Ladder	1 pt	Climb a 10-foot ladder at a 60-degree incline	Climbed to second step. Slipped sideways due to low friction on step surface, resulting in fall

7.1. Wall Drilling Task

We focus our first detailed discussion on the drilling task (see Table II), which exercised most of our system's components and required a full range of planning—from low-level end effector adjustment to more sophisticated constrained motion plans. The drilling task involved walking to a table, picking up a drill, turning it on, walking to a dry-board wall, cutting out a triangular section of the wall, and pushing the cutout free of the wall. The threshold for scoring any points was relatively high; each of the first two points was awarded for cutting one line segment of the triangle, and the third point required cutting the third segment and successfully knocking the triangle loose. Any cuts into the wall too far away from the triangle edges immediately voided that edge and eliminated the possibility of the third and fourth points. This resulted in most teams operating quite slowly and carefully.

Charts detailing task execution during the DRC Trials are illustrated in Figure 17, whose first row illustrates progress through the task over time. The initial phase of the task went smoothly and, after 3 min, the robot had reached for the drill and was ready to grasp. Reaching consisted of a scripted posture that raised the hand up and around the table, followed by a single reaching plan directly to a *sticky hand* attached to a model of the drill that had been fit by the perception operator. However, after reaching to the pregrasp position, the operator realized that the hand was unresponsive and would not close around the drill. During a three-minute delay, we quickly resolved the issue and grasped the drill. We learned after the event that the problem was due to wiring damage that caused intermittent electrical connectivity to the hand.

After grasping the drill, the operator commanded the robot to lower its arm and step to the side of the table. In our experiences in the lab walking with the drill and fire-hose in hand, we found that holding masses close to the torso gave improved stability. After walking up to the wall, the operators raised the drill into the head camera view using the posture planner, and refit the drill affordance precisely relative to the hand using the 3D fitting interface. From this point forward, the drill affordance was assumed to be fixed to the hand frame, and all drilling motions would be executed relative to the drill-bit frame. A plan was then generated to reposition the drill so that 1) the power button would be clearly in view of the camera, and 2) the button would be reachable by the right hand.

Using the pointer on the right hand and end effector teleoperation relative to the camera frame, the operator pressed the drill's power button. While this motion could easily be expressed by our affordance-based planning system, the combined imprecision of the forward kinematics and perception data made subcentimeter autonomous motions unreliable. During practice sessions, we verified that the drill had been turned on using audio continuously streamed across the network link from the onboard microphone. However, in the competition, background noise from the wind and crowd made it impossible to discern the sound of the drill motor. After realizing this, the operators captured a high-resolution image chip of the drill button to determine that the drill was on.

The perception operator then fit a triangular drill pattern affordance to the wall, with the corner points aligned to the RGB-colorized LIDAR view. After the operator clicked on each triangle vertex in the image, the system intersected the corresponding back-projected view rays with a wall plane that was automatically fit to the LIDAR data.

7.1.1. Drilling: Multistep Plan Generation and Validation

Executing the drilling motion (in a single sequence) was one of the more complicated planning capabilities we fielded in

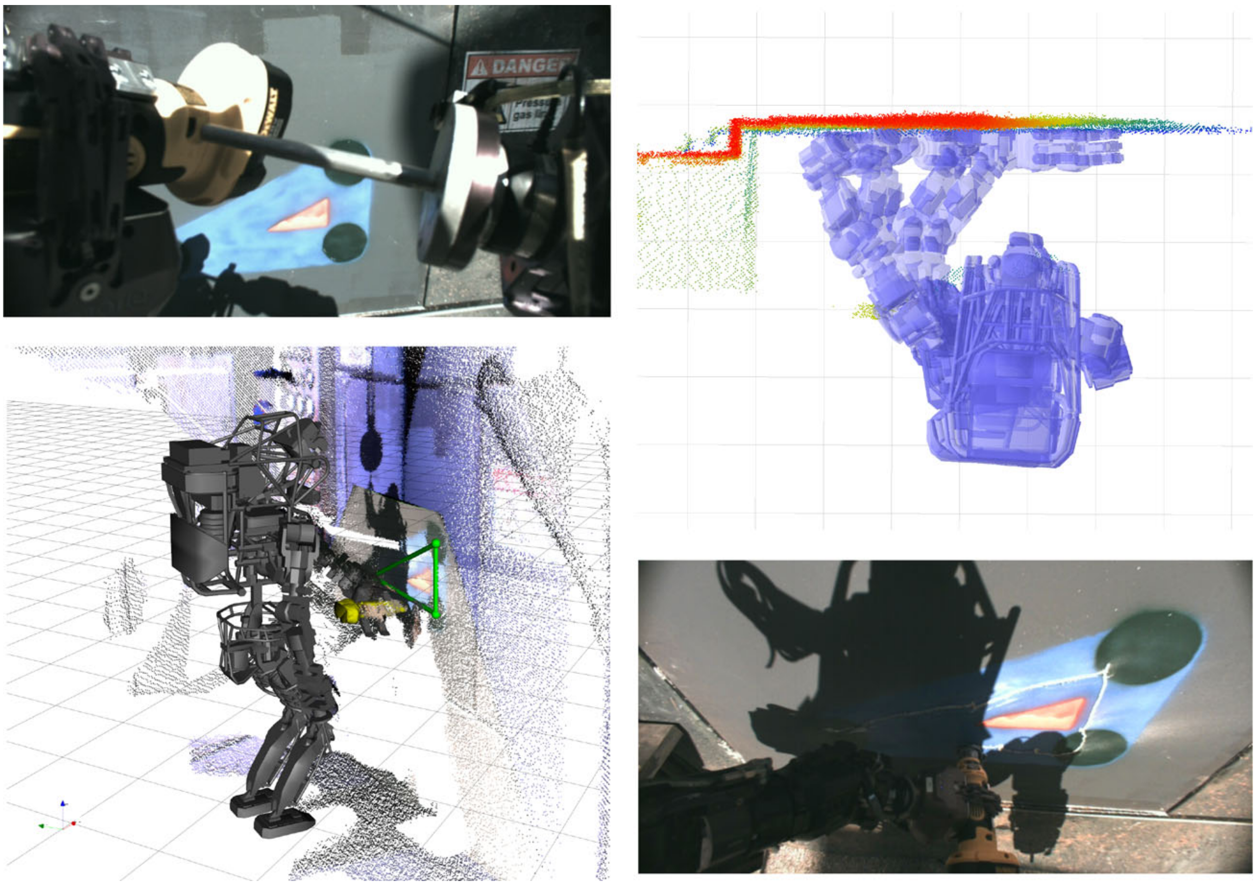


Figure 16. Top left: head camera view of the poker used to press the drill's power button. Top right: visualization of the nominal plan to reach all drill target points in the triangle. Bottom right: head camera view provides situational awareness to the operator as the robot executes the drilling plan. Bottom left: Robot avatar and RGB point cloud visualization with affordance models of the drill and triangle target.

the Trials. For complex, multistep manipulation tasks such as this, a greedy approach to plan generation has the possibility of failing midsequence, leaving the robot stuck in a state where the next step is kinematically infeasible. To address this issue, the planner attempted to ensure task feasibility when given a full task description (in this case the geometry of the triangle). Thus, before the robot even approached the wall, all kinematic constraints from the task were sent to the planner to generate a comprehensive motion. Such a plan is visualized in Figure 16. Included in the generated plan was an appropriate place to stand at the wall, which the planner then synthesized into a goal for the walking planner.

Once the robot reached the wall, the planner was queried again to generate a complete cutting motion that kept the tool axis perpendicular to the wall and at a constant depth inside the wall, and for which the robot would remain quasistatically stable. Like the previous multistep plan used to generate a walking goal, this motion was not executed as a single plan. However, its existence ensured the feasibility of the entire motion, and it was used as a seed to generate the subsequent smaller plans in the sequence.

7.1.2. Supervised Autonomous Plan Generation

The structure of the rest of the task lent itself to a straightforward sequence:

```
1: Move drill in front of first target, avoiding collisions with
   the wall.
2: Drill into first target
3: for Each line segment to cut do
4:   while Line segment is incomplete do
5:     Cut along line segment
6:     Refit triangular affordance
7:   end while
8: end for
```

Each robot motion above (lines 1, 2, and 5) required approval by the operator. The planner automatically corrected

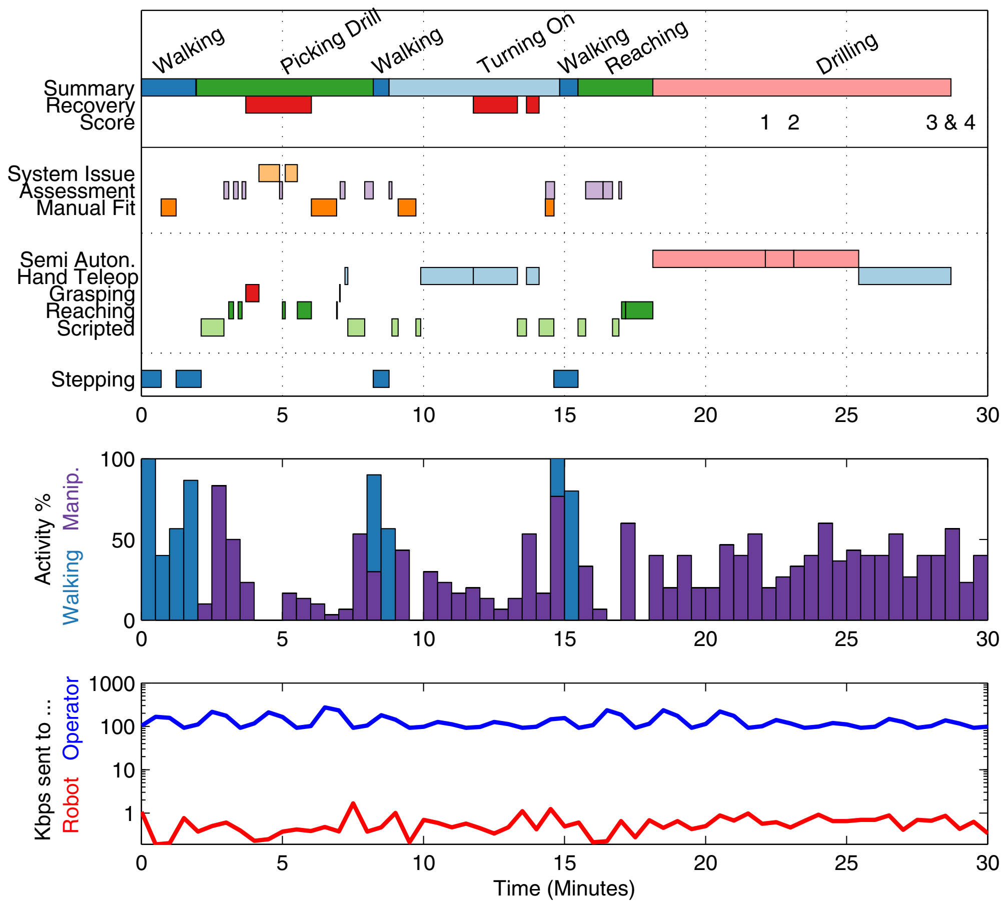


Figure 17. Analysis of the Drill Task. Top: Analysis of operating time. From top to bottom: summary of task components, time spent recovering from mistakes, and times when points were scored; time spent primarily stationary while the operators considered the situation and manually fit objects; manipulation modes; and time spent stepping. Center: Portion of time the robot spent moving integrated over 30 s intervals. Bottom: Data transmission rates from robot to operator (blue) and vice versa (red).

for deviations from the desired path, and transitioned to the next line segment when it judged the cut to be within the target region. Ultimately, however, the user supervised these autonomous plans and could, at any time, instead generate a teleoperated cutting motion on the plane of the wall.

A challenge while cutting the wall was maintaining an accurate state estimate in the presence of significant external forces and joint backlash. This uncertainty made shorter motions advantageous, since they minimized the risk of cutting outside the allowable regions. Manual refinements to the position of the triangle affordance (adjusted by sliding in the wall plane) also helped, and demonstrated a tight loop between (human) perception and robot control. The degree of autonomy embedded in the planner greatly increased the speed of execution, while still allowing for relatively short, safe cuts.

At the 18th minute, we commenced cutting and cut the entire pattern in a total of 8 min. After the edges were cut, we reverted to teleoperating the drill in the plane of the wall to push against the triangle, and finally, with a few minutes remaining, we dislodged it.

7.1.3. Analysis of Operation and Communication

As illustrated in Figure 17, we operated the robot in a variety of manners—from executing prescripted joint sequences to operator-supervised autonomy. In total, we executed 4 walking plans and 86 motion plans during the drill task. The vast majority of those were in semiautonomous mode (salmon colored section), in end effector teleoperation mode (blue), and in affordance-relative reaching. We aim to improve upon this in the coming year by implementing compliant motion control (e.g., in the case of pressure against the wall). Also illustrated is time spent executing scripted motions (light green). Other significant time blocks included resolving the systems issue (tan), assessing the next action of a plan (purple), and manual affordance fitting (orange).

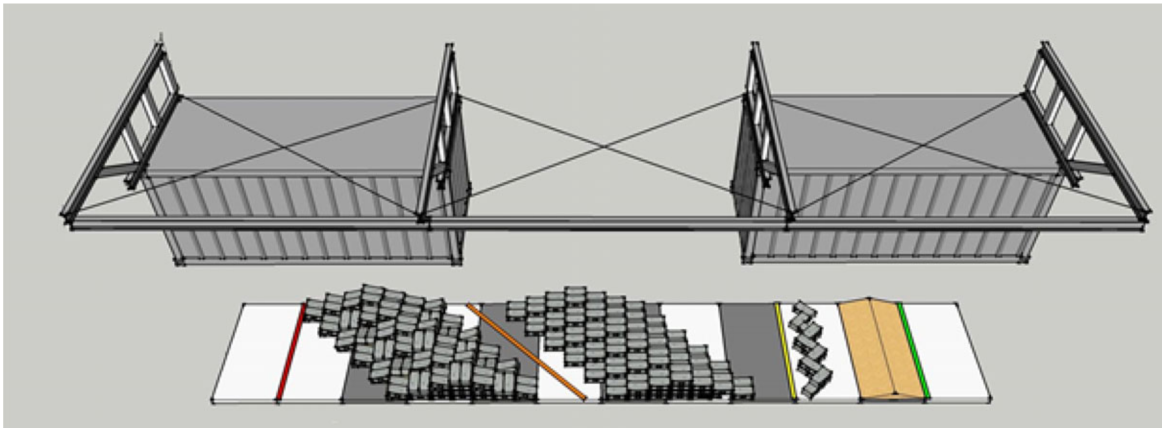


Figure 18. The DRC Trials Terrain Course (graphic credit: DARPA). Execution was from right to left. The cinder blocks on the left side were tilted at angles of 20–40 degrees.

The second plot in Figure 17 shows the percentage of time the robot was actively executing a plan, per 30-s period. Despite our operators' complete focus on the tasks, the majority of time was spent stationary, with the robot moving only 26% of the time (35% if walking is included). While more fluid user interfaces would likely increase these numbers, we believe significant gains can be achieved by building more autonomy into our system.

The lower plot in Figure 17 represents bidirectional bandwidth usage for the task. We can see that the usage was heavily one-sided, with incoming data from the robot (blue) outweighing plans communicated to the robot (red) by over two orders of magnitude. While not shown here, the configuration of the robot and its onboard sensors plus our own diagnostic information represented only 5% of incoming data—with the remainder being split between imagery and LIDAR. Note that the ripples in the data rate sent from the robot correspond to the 100 kbps/1 Mbps regimes alternating each minute.

These plots have several implications in the context of the networking parameters defined by DARPA that were intended to encourage autonomy. First, imposition of the 100 kbps/1 Mbps regimes required that each team implement downstream sensor data compression algorithms, but it did not encourage autonomy, as teams were free to transmit any number of commands upstream to the robot, provided the commands were sufficiently compact. Secondly, we found that the variable 50 ms/500 ms latency added in each of the alternating regimes had little effect on our operators' actions, which typically proceeded in the following sequence: perceive relevant features, create a motion plan, transmit the plan, then monitor execution while data are returned. The time taken to transmit any plan to the robot was dwarfed by the other actions—in particular, time spent *by the human* perceiving the surround and specifying and reviewing plans.

In summary, the bandwidth and latency fluctuations had little effect on our operation strategy. We also surmise that further reduction of the bandwidth (if applied uniformly) would not bring about an expected increase in autonomy—it would simply limit the operator to a sparser perception of the scene. Finally, it is our opinion that direct teleoperation, if implemented correctly, can be executed with much lower bandwidth requirements than our implemented semiautonomous system. For these reasons, we believe that autonomy would be more effectively incentivized through significantly longer latencies—on the order of 3–10 s—and by limiting the number of discrete commands that the operator can send to the robot.

7.2. Terrain Traversal Task

By comparison with the drilling task, the walking task was much more straightforward. The robot was required to cross a progressively more difficult course of cinder blocks in 30 min. The course is illustrated in Figure 18. We were one of only three teams to finish the course: IHMC completed the course with an Atlas robot and their own walking controller, while the SCHAFT robot also completed the course, executing a multistep walking gait much faster than the other teams. We did fall half-way through the task and required a restart (with a 5 min penalty), but we completed the course in the allotted time. (The robot fell due to contact between the robot's foot and the second part of the course, which may have been due to human error in placing a footstep goal.)

We utilized the Boston Dynamics stepping controller in the task, and we were the only Atlas-based team to complete the task using that controller. Use of Boston Dynamics' controller shifted our focus onto footstep placement. In the simulation phase of the competition, our footstep planning system typically calculated kinematically reachable

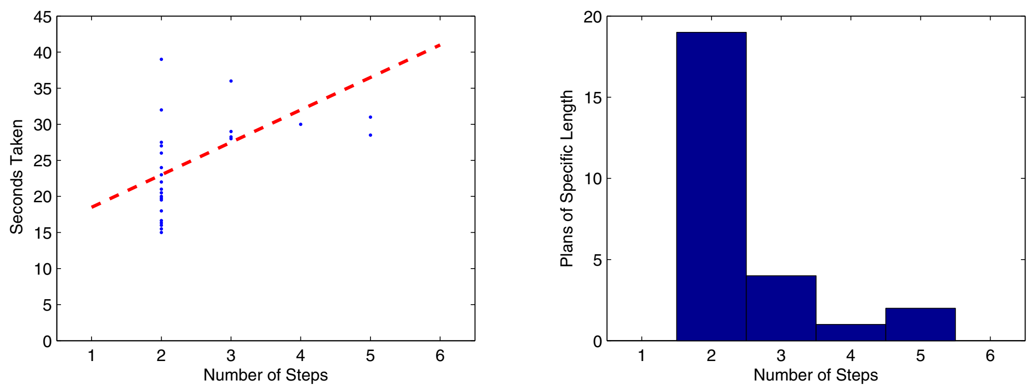


Figure 19. Analysis of Planning for the Walking Task. Left: Time taken to plan (blue dots) and execute (red dashed line) each footstep plan, for different plan sizes. Each blue dot represents the time taken for the operator to place a new set of steps on a terrain map. Right: Distribution of footsteps per walking plan. Typically we limited the robot to just two steps per plan due to state estimate drift.

footsteps around large obstacles or difficult terrain normals 10–15 steps at a time, as well as fitting foot normals to the sensed terrain. In the DRC Trials competition, the focus was instead on precise placement of pairs of footsteps as the margin of error during execution was very narrow.

While some teams built a template of the entire terrain course (with attached footstep locations) upon arriving at the Homestead Speedway, and while progressing carefully fit this template to a sensed LIDAR height-map, we made no such prior assumption so as to remain flexible. In retrospect, this meant that our progress was slower as each footstep had to be dragged into place along the sensed terrain map. It also meant that there was a significantly higher chance of operator error, which we attempted to minimize by having a second operator approve the footstep placements proposed by the primary operator. Once decided upon, these footstep placements were passed through to the Boston Dynamics controller for execution.

In pre-competition practice, we determined a series of heuristics which best ensured successful execution. The primary heuristic was that while stepping up and down on the cinder blocks, it was best to minimize the forward distance traveled by the robot because of the limited range of motion of the robot's legs—specifically the ankle. In fact, certain step patterns on the terrain course were impossible for the controller to execute, and we avoided paths across those patterns.

We also observed the drift rate of the robot's state estimate to be roughly 2 cm per step (enough for the robot's toe to clip a block edge). Although we investigated the suppression of this drift, as it is gait- and terrain-dependent, modeling was difficult. Instead, in the competition we planned and executed very short plans where drift would not endanger the robot.

Figure 19 shows a summary of time allocation during the walking task and the length of the executed plans. In total, 71 footsteps were required to complete the course, and these were divided into 27 separate plans. Two-thirds of the plans were two-step plans, which took our operator 22 s to create on average. As a result, almost exactly 50% of the time the robot was stationary while the the operator planned the next set of footsteps. In addition, the robot's average walking speed is much slower for shorter plans than longer ones: a two-step plan took 23 s to execute, and three-step plans took only four more seconds. This is because the robot's quasistatic gait spends significant time shifting its weight at the beginning and end of walking.

In summary, while our approach was successful, there is much room for improvement in a number of respects. We estimate that the Atlas robot continuously and dynamically walking across the terrain could cover the distance five times faster than the 25 min we required in the competition. We will discuss our future plans for this task in Section 8.3.

7.3. Human Operation versus Robot Autonomy

In the Trials, we presented a blend of human operation and autonomy. While we attempted to maintain an affordance-based approach at all times, this had to be balanced with the ability of the robot and our systems to react.

Even where our autonomous components were clearly beneficial, such as our object-fitting algorithms, a human eye or adjustment was always the last word. Where a slight orientation bias in an algorithm such as ICP could cause a subsequent grasp to fail, this bias would be corrected by the operator before planning reaches.

In the debris task, the operator mistakenly dropped a board under the robot's feet by failing to reason about its impact with the ground, but by comparison our autonomous system could not reason about a collision between a board held in hand and the rest of the debris pile during a retraction. In the drilling task, one of the drills would stop drilling after a short period of time and require a retrigger—something that is difficult to detect autonomously but perhaps easier to avoid by frequently regrasping. Finally, in the previous section we detailed the number of manual foot placements required to complete the terrain course—which is neither reproducible nor efficient, yet automated placement requires reasoning in a high-dimensional space.

Thus, we assert that while autonomous systems can avoid errors and produce repeatable performance, the threshold for their use is often high. In Section 8.3, we discuss some of our plans to automate pertinent subtasks for the next phase of the competition.

8. DISCUSSION

8.1. Limitations and Lessons Learned

From our experience thus far, we have learned lessons along a number of dimensions: about the precision and accuracy with which we can control Atlas; about the viability of our high-level approach; about the operator burden implied by our approach; about the importance of planning for future (not just imminent) actions; and about our team organization.

When we formulated our high-level approach in early 2012, we anticipated being able to achieve reasonable kinematic precision and accuracy, i.e., that configuration space position tracking would yield accurate task space trajectory tracking. This assumption held during the VRC, since the model was kinematically and dynamically perfect and sensor noise was negligible or absent. However, when the Atlas arrived in August 2013, we quickly learned that the provided model was not as faithful to the physical robot, and that achieving centimeter-accurate end effector placement in the face of significant joint friction and backlash would be a significant challenge. Our approach in the DRC Trials was to address this issue with a combination of careful kinematic calibration, high-gain position control, and basic compensators for friction and backlash effects, which we were able to model quite accurately. Moving forward, we intend to do closed-loop task-space control with the goal being to achieve very good end effector positioning with a more dynamic and compliant robot.

On the perception side, the sensor head provided with the robot contained a high-quality stereo camera system paired with an FPGA that computed dense depth images. We did not use this information because 1) stereo depth was expected to be unreliable in the field, and 2) without a neck yaw degree of freedom, we had no expectation of a visual line of sight in many of our tasks. Our approach relied heavily on LIDAR range images that provided updates to the operator below 1 Hz. Inevitably this precluded high-rate camera-to-hand coordination, with the robot's actions limited to closed-loop execution of kinematic sequences.

We had expected that generating robust walking on uneven terrain estimated from perception data could pose a significant challenge. However, this was not the case in either simulation or on the real robot. The quality of the height map estimates obtained from the laser-based perception system was exceptional, and the walking controllers worked with relatively little adaptation.

The decision to use MATLAB for a significant portion of our code base was highly controversial within the team. Each planner—and even the controller on the field computer—ran as a separate (desktop) MATLAB instance during the actual competition. The performance of these systems, after optimizing MATLAB code and porting the most expensive calculations to mex C++, proved to be sufficient for real-time operation, and we were able to push sufficient rigor into the software development process by adding our own MATLAB type-checking and unit tests. In the end, the decision to keep a high-level modeling language central in our work-flow allowed us to prototype and integrate new planning and control capabilities very rapidly, and we believe now that it would have been difficult to achieve as much solely in a lower-level language.

While developing task story boards and architecture for the system, we assumed that various subfunctions such as object tracking, and composition of low-level actions into action "macros," could be straightforwardly implemented. As we proceeded to implementation, it became clear that in fact full automation would be more difficult than anticipated, so we proceeded to relegate these functions to the human operators through additional interface elements. We learned that this path leads to overburdening the operators with an overly complex and specialized interface, and a long list of steps to take during task execution. For example, at the DRC Trials we needed a designated perception operator to periodically adjust the fit of affordances to sensor data. As we develop our system for the DRC Finals, we are trying to be much more careful about allowing complex elements to creep into the interface, and trying to avoid simply assuming that the operator can pick up whatever slack the system leaves.

Our initial DARPA proposal envisioned an interface providing fluid visualization of past plans and actions, current state, and a variety of future plans to choose among. Our implementation, however, visualizes only the current state and a single plan, which can be confirmed, adjusted, canceled, or aborted while in progress (provided the network allows it). Moreover, our system as of the DRC Trials enables the operator to command the system only with respect to its current state, not with respect to some future (i.e., planned or anticipated) state. We found that this forced us to watch the competition clock run down as the robot completed some long action (such as moving a piece of debris to the side, dropping it, and bringing its hand back for a new grasp), even though we already had good plans in mind for the next several steps. For the DRC Finals, we intend to remove this limitation, so that we can input and consider various plans for future actions even as the robot is carrying out some past command. The system will use a controller with guards to achieve serial execution of chained plans, with pauses and a return to operator control in case of plan execution failure.

A number of our longer-term project plans were too ambitious to be achieved in time for the December Trials. Those plans included: a state estimator integrating kinematic, inertial, and exteroceptive (visual and LIDAR) data; whole-body force control for locomotion and manipulation; a fully automated object tracker coupled to the state estimator; and motion capture-based upper-body teleoperation as a backup strategy in case our affordance-based strategy was insufficient. Although we were unable to completely implement any of these components in the four months we had the robot, our system was able to function effectively at the Trials even in their absence, as we were able to compensate for missing functionality by adding load to the operating team. As the Finals approach, we plan to reverse the process, i.e., to shed load from the operators by bringing each of the pieces above to partial or full functionality.

We also learned that when a human is involved in the system, its operation will necessarily be imperfect. Humans make mistakes, and some of those mistakes will manifest themselves as undesirable robot behavior. While it is easy to focus on achieving flawless execution of complex motion plans, we learned that the ability to quickly recover from and continue after errors or failure was perhaps more important for fielding a robust solution with a human in the loop.

Finally, we learned to practice fluidity in our team organization. At the outset, we had organized by functional subsystem: perception, planning, control, interface, network, infrastructure. But as the DRC Trials approached, we reminded ourselves that our end-to-end success metric was the number of points scored, and that we needed to score points on each of the eight tasks. We restructured on a per-task basis with a different operator for each (Fallon for debris and fire-hose, Kuindersma for valve, Deits for terrain and door, Valenzuela for ladder, Posa for drill and driving), while asking the subsystem leads to continue in those roles, effectively serving double duty.

8.2. Reflection on the Competition

We were greatly impressed with the organization, definition, and execution of the DRC Trials competition by DARPA. Despite some early networking hiccups, the entire competition ran without major technical issues. Given the uncertainty about the capacity of the robots to achieve the tasks, DARPA carefully modulated each task's difficulty to facilitate at least partial execution by the majority of teams.

Basic issues such as the amount of information given to the teams about each particular task, or even the strength of the wind on a particular day, had a huge effect what the robots were capable of. We must remember that humanoid technology is only just at the stage of being practically useful outside controlled lab environments. As such, there are a number of variables that can be carefully modulated to shepherd the next set of advances. Our own goals are to increase the level of autonomy (to speed up task execution *and* to reduce the burden on the human operator), to increase the dynamism achieved when executing motions, and to decrease the amount of time the robot spends idle between motions.

The speed of execution seen in the DRC Trials was largely dictated by the fact that teams were given only a single chance to perform each task, with a very high cost for interventions (costing 5 min of the 30 min allotted to each task). Furthermore, the scoring prioritized completion over the time taken and over the operator's cognitive load.

Another feature of the DRC Trials was that the task descriptions prior to the competition provided detailed information about the physical task layouts, which allowed teams (including ours) to practice with replica layouts under the assumption that certain actions refined during practice would work in competition. Effectively, the majority of human planning had been carried out in advance of the competition. For these robots to be truly useful in future, we must design systems that, rather than depending on prior knowledge, can be brought to bear fluidly on novel problems.

These observations suggest the need to develop flexible robot autonomy with the operator moving to a supervisory role rather than directing each action of the robot. As discussed in Section 7, however, it is not obvious which communication constraints DARPA should impose in order to encourage greater autonomy. Simply limiting available bandwidth and enforcing minimum latency are evidently insufficient, at least for the Trials tasks, since many teams were able to score highly using direct teleoperation.

We observe that, generally speaking, the less direction needed by a *human* subordinate, the more independent (i.e., autonomous) that subordinate is considered to be. We suggest that for the DRC Finals, DARPA adopt a scoring system that incentivizes independent operation by the robot. For example, DARPA could use its existing network infrastructure to count successive transmissions from the operator(s) to the robot, applying a scoring debit for each transmission.

8.3. Future Plans

We are actively porting our whole-body dynamic balancing and walking controller developed for the VRC (Kuindersma, Permenter, & Tedrake, 2014) to Atlas. We expect this to have several practical benefits: greater flexibility in executing whole-body manipulation plans, more robust stabilization in the face of perturbations (such as wind and foot slip), and faster locomotion while maintaining precise footstep placement.

In contrast to the position-based control used for the Trials, we will use force control. This will allow us to, e.g., more easily achieve compliant interaction with the environment for applying forces against a wall or door, or aligning a hose to a standpipe. One of the primary challenges to achieving high-performance model-based control is capturing a sufficiently accurate model of the system. We have therefore placed emphasis on developing algorithmic and experimental methods to estimate the inertial and frictional parameters of the robot. We have performed a series of initial balancing, walking, and compliant upper-body control experiments that have so far given promising results.

As discussed in Section 7.2, we intend to improve on our ability to traverse uneven terrain. In addition to faster locomotion, we are developing algorithms to enable automated footstep planning by interpreting the problem as a series of convex optimizations on the set of obstacle-free spaces (Deits & Tedrake, 2014). We have also had initial progress in extending the number of steps that can be continuously executed by integrating LIDAR as part of our state estimator to reject position drift. This has enabled the Atlas robot to traverse parts of the terrain course three times more quickly than previously.

We are extending our whole-body planning methods in several ways. We are incorporating self-collision-free motion into our planning system and are currently focused on developing a method for quickly producing collision-free motions for manipulation in clutter (for example, the debris pile). We are also exploring new dynamic motion planning algorithms that move beyond the kinematic planners used in the Trials to generate dynamically feasible locomotion and manipulation trajectories.

To reduce the burden on the human operator and to be able to perform useful work through periods of total communication loss with the robot, we are working toward a system that places greater emphasis on task-level autonomy. For example, the human operator will be able to provide a state-machine structure for a task, e.g., "walk to the drill, grasp the drill, walk over to the wall, turn on the drill, . . . ," and predicates that allow the robot to determine, e.g., when a grasp has failed or succeeded with some probability, allowing it to make reasonable decisions about what the next action should be. An initial demonstration of an autonomous walking, drill localization, and grasping sequence yielded a factor of 2 decrease in the time needed to achieve the result, compared to a highly interactive approach.

9. CONCLUSIONS

This paper described the key ideas behind the system developed by Team MIT to compete in the DRC Trials in December 2013.

We have presented an overview of the robot and its sensor suite, with a particular focus on how its design influenced the execution of the DRC Trials tasks. The major challenge of calibrating this 28-DOF robot and its sensors was discussed, as our approach relies upon a significant degree of kinematic accuracy.

Subsequently, an overview of our software system was presented. We developed a stateless perception subsystem that was paired with an efficient data transmission module to make the robot's internal state available to our operators within the communication constraints of the competition.

At the operator station, our system utilized robust semi-automated fitting algorithms that were developed with a focus on competition reliability. Using a few simple gestures, our operator could precisely fit primitive objects of interest. Having framed the locations of these objects (or "affordances"), our planning system was commanded via a flexible user interface to compute motion sequences that would enable the robot to engage affordances on their functional level: turning valves, opening doors, and operating drills by exercising the degrees of freedom of their underlying kinematic models.

While such an approach seemed straightforward in the simulation phase of this competition, maintaining a high -level approach with the real robot has been difficult. We have had insufficient sensing accuracy and dexterity to, for example, mate a hose. Limitations in our system's ability to reason about a variety of physical factors made us slow and clumsy to react at times. However, in certain tasks the affordance approach has been successful, and we continue to develop and expand this central capability.

We feel that our performance at the Trials competition validates our approach. We aim to expand upon this work in the coming year to achieve more dynamic, more autonomous, and more capable robot performance in the DRC Finals.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency via Air Force Research Laboratory award FA8750-12-1-0321, and the Office of Naval Research via award N00014-12-1-0071. We are also grateful to the team's many supporters both inside and outside MIT (listed at http://drc.mit.edu), including our families and friends.

We are also grateful to Boston Dynamics, Carnegie Robotics, the Open Source Robotics Foundation, iRobot Corporation, and Sandia National Laboratories for their support during the DRC.

REFERENCES

Alunni, N., Phillips-Grafflin, C., Suay, H. B., Mainprice, J., Berenson, D., Chernova, S., Lindeman, R., Lofaro, D., & Oh, P. Y. (2004). DARPA robotics challenge: Towards a user-guided manipulation framework for high-DOF robots. In

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Chicago.

Buss, S. R. (2004). Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods. IEEE Transactions in Robotics and Automation, 17.

Casper, J., & Murphy, R. (2003). Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. IEEE Transactions on Cybernetics, 33(3), 367–385.

Chestnutt, J., Kuffner, J., Nishiwaki, K., & Kagami, S. (2003). Planning biped navigation strategies in complex environments. Karlsruhe, Germany.

Deits, R. L., & Tedrake, R. (2014). Computing large convex regions of obstacle-free space through semi-definite programming. In Workshop on the Algorithmic Fundamentals of Robotics (WAFR), Istanbul, Turkey.

Dellin, C., Strabala, K., Haynes, G. C., Stager, D., & Srinivasa, S. (2014). Guided manipulation planning at the DARPA robotics challenge trials. In International Symposium on Experimental Robotics (ISER), Marrakech.

Gibson, J. J. (1977). The theory of affordances. In Shaw, R., & Bransford, J. (eds.), Perceiving, acting, and knowing. Hillsdale, NJ: Lawrence Erlbaum Associates.

Gill, P. E., Murray, W., & Saunders, M. A. (2002). SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM Journal on Optimization, 12(4), 979–1006.

Hartley, R., & Kang, S. B. (2007). Parameter-free radial distortion correction with center of distortion estimation. IEEE Transactions in Pattern Analysis and Machine Intelligence, 29(8), 1309–1321.

Huang, A., Olson, E., & Moore, D. (2010). LCM: Lightweight communications and marshalling. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan.

Huang, W., Kim, J., & Atkeson, C. (2013). Energy-based optimal step planning for humanoids. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 3124–3129), Karlsruhe, Germany.

Kemp, C. C., Fitzpatrick, P., Hirukawa, H., Yokoi, K., Harada, K., & Matsumoto, Y. (2008). Humanoids. In Siciliano, B., & Khatib, O. (eds.), Springer Handbook of Robotics (pp. 1307–1333). Berlin-Heidelberg: Springer.

Khalil, W., & Dombre, E. (2004). Modeling, identification and control of robots. Amsterdam: Elsevier.

Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M., & Inoue, H. (2001). Footstep planning among obstacles for biped robots. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (vol. 1, pp. 500–505), Maui, Hawaii.

Kuindersma, S., Permenter, F., & Tedrake, R. (2014). An efficiently solvable quadratic program for stabilizing dynamic locomotion. In IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China.

Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., et al. (2008). A perception-driven autonomous urban vehicle. Journal of Field Robotics, 25(10), 727–774.

Matsuno, F., Sato, N., Kon, K., Igarashi, H., Kimura, T., & Murphy, R. (2014). Utilization of robot systems in disaster sites of the Great Eastern Japan Earthquake. In Yoshida, K., & Tadokoro, S. (eds.), Field and Service Robotics, volume 92 of Springer Tracts in Advanced Robotics (pp. 1–17). Berlin-Heidelberg: Springer.

Michael, N., Shen, S., Mohta, K., Mulgaonkar, Y., Kumar, V., Nagatani, K., Okada, Y., Kiribayashi, S., Otake, K., Yoshida, K., Ohno, K., Takeuchi, E., & Tadokoro, S. (2012). Collaborative mapping of an earthquake-damaged building via ground and aerial robots. Journal of Field Robotics, 29(5), 832–841.

Sheh, R., Jacoff, A., Virts, A.-M., Kimura, T., Pellenz, J., Schwertfeger, S., & Suthakorn, J. (2014). Advancing the state of urban search and rescue robotics through the RoboCupRescue robot league competition. In Yoshida, K., & Tadokoro, S. (eds.), Field and service robotics, volume 92 of Springer Tracts in Advanced Robotics (pp. 127–142). Berlin-Heidelberg: Springer.

Sucan, I. (2014). ROS URDF package. http://wiki.ros.org/urdf (last accessed: Feb 2014).

Tedrake, R. (2014). Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems. http://drake.mit.edu.

Tedrake, R., Fallon, M., Karumanchi, S., Kuindersma, S., Antone, M., Schneider, T., Howard, T., Walter, M., Dai, H., Deits, R., Fleder, M., Fourie, D., Hammoud, R., Hemachandra, S., Ilardi, P., Pérez-D'Arpino, C., Pillai, S., Valenzuela, A., Cantu, C., Dolan, C., Evans, I., Jorgensen, S., Kristeller, J., Shah, J. A., Iagnemma, K., & Teller, S. (2014). A summary of Team MIT's approach to the Virtual Robotics Challenge. In IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China.

Yamauchi, B. (2004). PackBot: A versatile platform for military robotics. In Gerhart, G. R., Shoemaker, C. M., & Gage, D. W. (eds.), Proceedings of SPIE 5422, Unmanned Ground Vehicle Technology VI (pp. 228–237).