Team THOR's Entry in the DARPA Robotics Challenge Finals 2015

Stephen G. McGill and Seung-Joon Yi
GRASP Lab, University of Pennsylvania, Philadelphia, Pennsylvania 19104
e-mail: smcgill3@seas.upenn.edu, seungjoon.yi@gmail.com
Hak Yi
School of Mechanical Engineering, Kyungpook National University, Daegu, South Korea 41566
e-mail: yihak@knu.ac.kr
Min Sung Ahn, Sanghyun Cho, Kevin Liu, and Daniel Sun
RoMeLa Lab, University of California at Los Angeles, Los Angeles, California 90095
e-mail: aminsung@ucla.edu, albe1022@ucla.edu, kevinliu676@gmail.com, danielsun@ucla.edu
Bhoram Lee, Heejin Jeong, and Jinwook Huh
GRASP Lab, University of Pennsylvania, Philadelphia, Pennsylvania 19104
e-mail: bhorlee@seas.upenn.edu, heejinj@seas.upenn.edu, jinwookh@seas.upenn.edu
Dennis Hong
RoMeLa Lab, University of California at Los Angeles, Los Angeles, California 90095
e-mail: dennishong@ucla.edu
Daniel D. Lee
GRASP Lab, University of Pennsylvania, Philadelphia, Pennsylvania 19104
e-mail: ddlee@seas.upenn.edu

Received 2 October 2015; accepted 11 July 2016

This paper describes Team THOR's approach to human-in-the-loop disaster response robotics for the 2015 DARPA Robotics Challenge (DRC) Finals. Under the duress of unpredictable networking and terrain, fluid operator interactions and dynamic disturbance rejection become major concerns for effective teleoperation. We present a humanoid robot designed to effectively traverse a disaster environment while allowing for a wide range of manipulation abilities. To complement the robot hardware, a hierarchical software foundation implements network strategies that provide real-time feedback to an operator under restricted bandwidth using layered user interfaces. Our strategy for humanoid locomotion includes a backward-facing knee configuration paired with specialized toe and heel lifting strategies that allow the robot to traverse difficult surfaces while rejecting external perturbations. With an upper body planner that encodes operator preferences, predictable motion plans are executed in unforeseen circumstances. These plans are critical for manipulation in unknown environments. Our approach was validated during the DRC Finals competition, where Team THOR scored three points in 18 min of operation time, and the results are presented along with an analysis of each task. © 2016 Wiley Periodicals, Inc.

1. INTRODUCTION

The recent DARPA Robotics Challenge (DRC) Finals required a complete robotic system that can manipulate human tools and move about in an unstructured environment. The Finals incorporated a set of eight consecutive manipulation challenges outdoors on rough pavement. We developed the THOR-RD (Tactical Hazardous Operations Robot–Rapid Development) robot, shown in Figure 1, as an evolution of the hardware from the Trials, to compete in the challenge. With it, we developed a versatile software platform for planning and locomotion. The competition also focuses on high-performance human robot interfaces (Yanco et al., 2015) as the link established between the robot and a human operator underwent varying network conditions, including blackout periods.

The variability in the communication channel requires a corresponding variability in the teleoperation control mechanisms. Sliding between low-level and high-level control is a target for development of many DRC teams (Murphy, 2015), typically understood in the context of semiautonomy (Heger & Singh, 2006). Semiautonomous behavior becomes a critical aspect for disaster response, where the robot agent can observe local information with high fidelity,

Direct correspondence to: Seung-Joon Yi, e-mail: seungjoon.yi@gmail.com

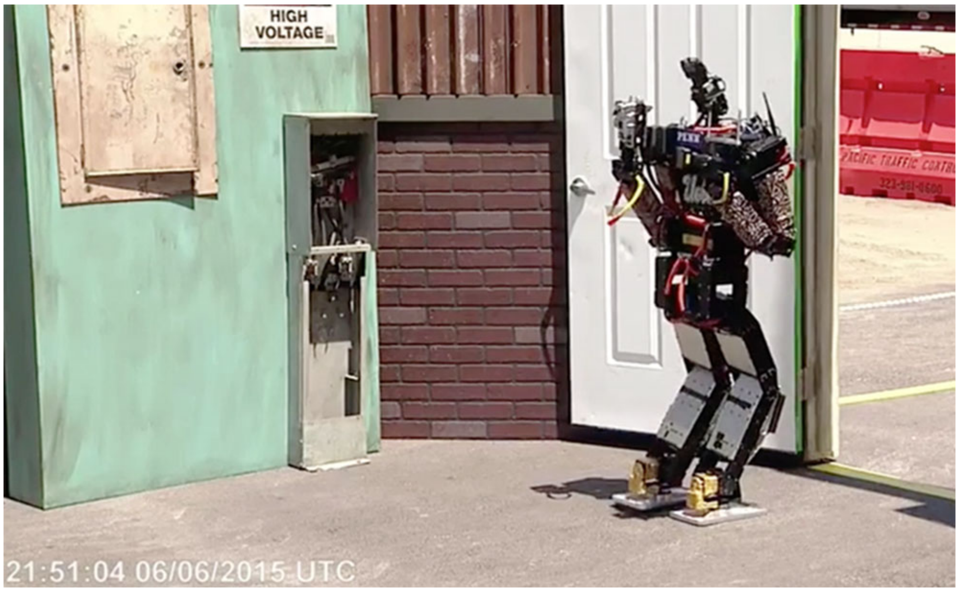


Figure 1. THOR-RD enters the indoor environment after opening the door in the DRC Finals using the backwards-knee configuration.

but the remote operator can only furnish a representation in their mind.

When deploying high degree of freedom (DOF) robots in unknown environments, human operators are often faced with challenging edge case conditions. Robustness in the face of uncertainty, a problem throughout robotics research history (Slotine, 1985), becomes even more important in disaster scenarios. Motors can break, practiced arm plans can fail, and the terrain can prove more difficult than imagined. Algorithms that are nimble enough to recover and adapt become major requirements for disaster response robots (Burke, Murphy, Rogers, Lumelsky, & Scholtz, 2004).

In this work, we present our approach for the DRC Finals, which has been extended in many ways since the DRC Trials (Yi et al., 2014). The hardware is more powerful and more robust to endure much longer test runs without human intervention. The locomotion controller is improved so that the robot can walk over unstructured surfaces while rejecting external perturbations. The arm controller is generalized so that it can be used for totally unknown tasks that require a large workspace. Finally, the communication and remote operation software is designed to handle a bandwidth-throttled link with blackouts.

The paper proceeds as follows. Section 2 describes the hardware platform we used for the DRC Finals. Section 3 describes the software for networked communication and human-robot interaction. Section 4 explains the manipulation framework, and Section 5 discusses the locomotion controller, rough terrain negotiation, and full-body behaviors. Section 6 presents results from the DRC Finals held in June of 2015. Finally, we conclude with lessons learned and a discussion of future work.

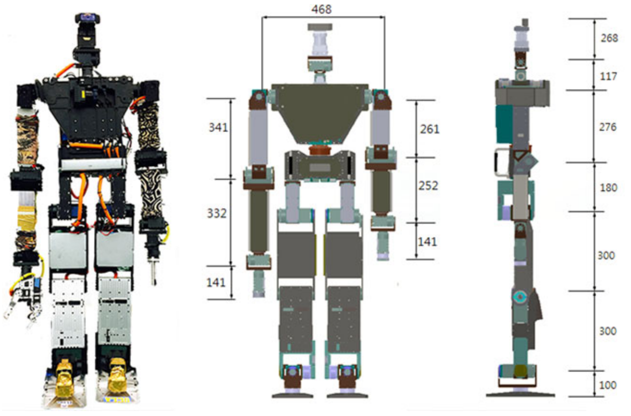


Figure 2. With its dimensions (shown in millimeters), the THOR-RD robot represents a large-scale humanoid robot that can work in areas designed for adult humans.

2. HARDWARE ARCHITECTURE

THOR-RD, shown in Figure 2, is a full-sized humanoid robot that stands 1.5 m tall and weighs 54 kg, with a wingspan of 1.95 m. THOR-RD represents an upgrade in several ways from the THOR-OP of the DRC Trials. It has 33 degrees of freedom (DOFs), with 7 DOFs in each arm, 6 DOFs in each leg, 3 DOFs in one hand, 2 DOFs in the waist, and 2 DOFs in the neck. As in the DRC Trials, the hardware of THOR-RD is composed of modular actuators and standardized structural components, which makes it easy to test different configurations and service damaged components.

The biggest improvement over the THOR-OP platform from the DRC Trials is a redesigned leg, where the knee

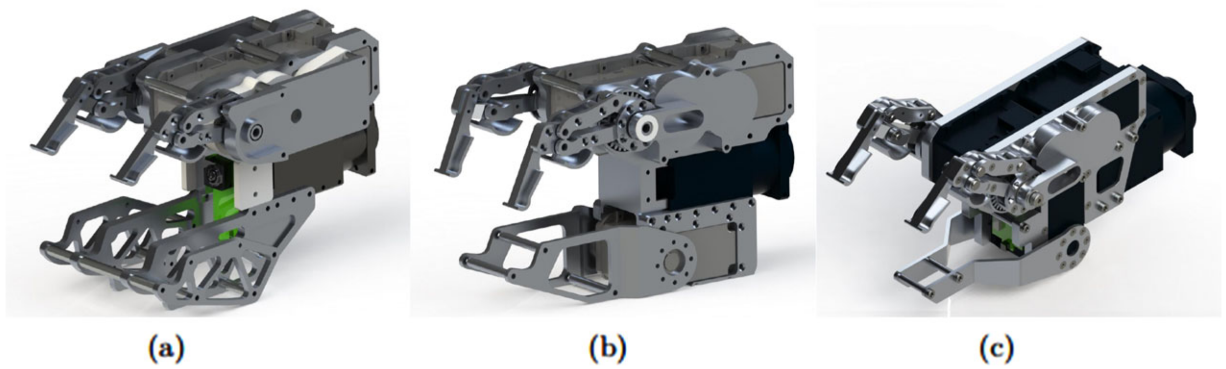


Figure 3. The gripper design was iterated to reduce complexity and weight while allowing for general grasps. (a) Two-fingered gripper with a fixed palm. (b) Modular three-fingered gripper. (c) Final version with a slim profile to minimize self-collision.

joint is powered by two custom actuators to have sufficient torque for standing up from the ground and traversing uneven terrain. The hip joints are redesigned to ensure a wide range of motion while reducing self-collision. Finally, previously exposed data and power cables are rerouted cleanly to minimize the risk of losing connection.

As a commercial product, the THOR series robot was adopted by a number of teams for the DRC Finals competition, but we heavily customized our version according to our design philosophy. Our customizations include end effectors, arm dimensions, electrical interface modules, and minimal sensors. The most unique aspect of our platform is the asymmetric arm setup. One arm is equipped with an actuated gripper and the other arm has a passive hand consisting of two metal rods. To keep the distances from shoulder to end effectors roughly the same for each arm, we use asymmetric arm dimensions as well. The longer arm with the passive end effector includes a shorter upper arm and lower link lengths.

2.1. Gripper

We used lightweight custom-made grippers with Dynamixel actuators for both the DRC Trials and Finals. At the Trials, we deployed a two-finger gripper with a fixed palm. Each finger is an underactuated two DOF four-bar linkage that is able to conform around a wide range of objects with secure grips (Rouleau & Hong, 2014). In addition, we used modular wrist attachments with various task-specific passive appendages such as a rod or a hook.

For the DRC Finals, field operators can no longer reconfigure the gripper between tasks. A versatile yet robust hand is required to handle all the given tasks while surviving contacts with objects throughout an entire run. Because it is hard to design an actuating hand that is both lightweight and robust against impacts, we decided to adopt an asymmetric end effector setup. We use a lightweight active gripper in one arm and a robust passive hand in the other, and only use the active gripper when the grasping is crucial for the completion of the task. As we cannot use task-specific appendages anymore, the active gripper needs to be more versatile. The main shortcoming of the previous gripper is that it has only two active fingers in one side, so the palm must be aligned precisely with the gripping object to secure the object with full force. Also, as only one side of the hand has actuating fingers, it is hard to pick up large objects such as the wooden pieces in the Debris task.

We designed a new three-finger gripper with two main goals. It should be able to grab a wide range of objects while tolerating some positioning error, and it should be lightweight with a short gripping position to keep the wrist actuator load low. The iterations of the new gripper are shown in Figure 3. The initial design utilizes a modular finger design that uses the wrist yaw actuator as a structural component of the assembly, with each of the modules attaching to it. The modularity greatly helped in finding the optimal location of fingers from iterated testing with prototype hands.

2.2. Foot

The foot is the main contact point of a humanoid robot, and its properties, such as geometry and sole material, can greatly affect the stability of the robot. For this reason, some humanoid robotics competitions, such as RoboCup, set limits on the foot size and the center of mass height of the robot to keep the bipedal locomotion challenging. The DRC, on the other hand, has no such rules and it even allows for statically stable nonhumanoid robots, so there is no reason not to equip the robot with large feet for more stability.

Still, most humanoid robots have fairly small feet because they look more natural and they help in traversing uneven terrain by increasing possible footholds. Smaller feet require smaller stride lengths to step between different surfaces. The original THOR-OP robot uses a relatively small foot with a thick footsole that works well for flat surfaces and is necessary for uneven terrain because the THOR robots have fairly short leg dimensions compared to

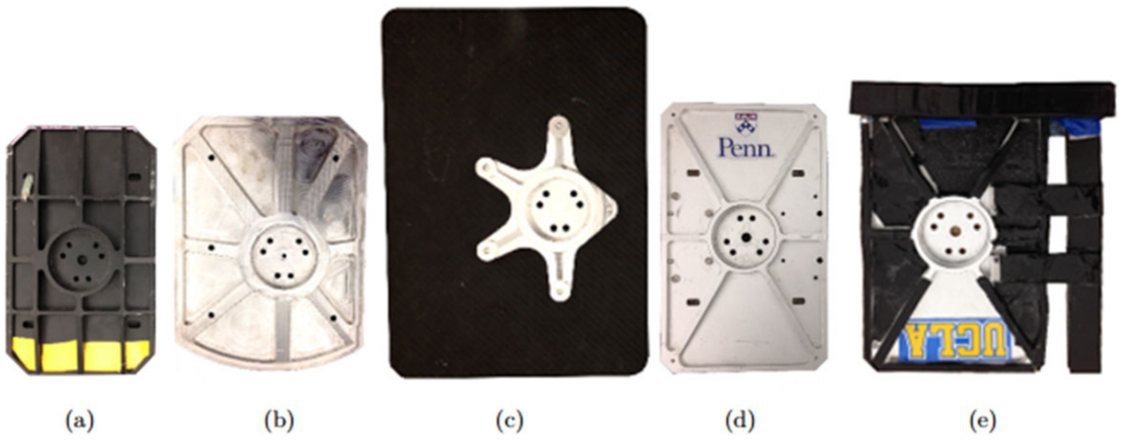


Figure 4. Foot designs for the THOR-RD robot needed to balance a large support region with mobility concerns. (a) Stock THOR-OP foot. (b) Prototype foot with wide support area and internal damper. (c) Large-sized foot made of carbon fiber to reduce weight. (d) Stock THOR-RD foot. (e) Adjustable foot with bolt on supports.

obstacle sizes. However, the DRC Finals require the utmost stability from the robot because the robots can be seriously damaged from a single fall due to the lack of safety measures. Some teams have even decided not to walk most of the time; we have decided to look for foot design changes that keep all the functionality of the robot while making it more stable.

We tried a number of foot design iterations, shown in Figure 4. We prototyped a big foot that almost fully covers the cinder blocks of the rough terrain. Such a large foot hampered locomotion performance due to timing issues upon landing and the high leg inertia. We also were worried about the possibility that the gas pedal area of the Polaris vehicle would pose collision issues with large feet. The final design incorporates additional support along the foot edges. The total width and height of the foot can be adjusted easily on the field by remounting the additional support. Unfortunately, we found that the Polaris pedal area leaves little room for foot size expansion, and did not use oversized feet or bolt on supports. In the future, we would like to provide quantitative measurements to rigorously guide our design decisions.

2.3. Sensors and Electronics

Our main design policy for the THOR-RD platform is to keep it simple and reliable, so we keep the sensor suite nearly unchanged from the Trials. A Microstrain 3DM-GX4-25 inertial sensor provides raw accelerometer and gyro data along with filtered pose estimates. Four independent RS485 chains provide communication to the four chains of motors via a USB interface. Two ATI Mini58 force-torque sensors on the feet aid balancing algorithms. One Logitech C920 HD Pro USB webcam mounted on the head captures audio and video. For more situational awareness, the left wrist is equipped with a Logitech C905 Webcam. For depth perception, two Ethernet-based Hokuyo UTM-30LX-EW LIDAR sensors are utilized. A servo motor pans a vertically mounted LIDAR in the chest. A head-mounted LIDAR provides 2D localization cues, and is moved by the head actuators.

To diversify sensing channels for perception, our team tested the usability of the Kinect 2.0 RGBD sensor for providing two-dimensional (2D) colored depth readings. The Kinect 2.0 works based on the time of flight principle (Fankhauser et al., 2015), and our extensive outdoor tests yielded promising results. Even though most pixels provided meaningless noise under bright sunlight, the center area gave reasonable readings for nearby distances (< 0.8 m). In general, RGBD sensors have an advantage compared to the LIDAR scanner. They provide 2D depth information at a similar rate to the LIDAR, which provides 1D depth information. The alignment between depth and color information is relatively well established, which is another merit. The sensor was omitted from the Finals due to weight and power supply concerns, however.

THOR-RD's computer and sensors run on a portable 12 V 120 Wh lithium ion battery that provides more than 3 hours of continuous operation in practice. The motors are powered with two 24 V 488 Wh lithium polymer batteries, which are doubled from the 288 Wh batteries we used for THOR-OP. This big battery increase ensures continuous operation for over an hour in the worst case. In practice, the robot motor system consumes less than 250 W during walking, and the batteries provide enough energy for hours of testing.

Instead of two 1.6 GHz AMD computers in THOR-OP for the Trials, we use a single Core i7 Haswell NUC computer for onboard processing. The Intel CPU provided more than twice the computation ability with half the power

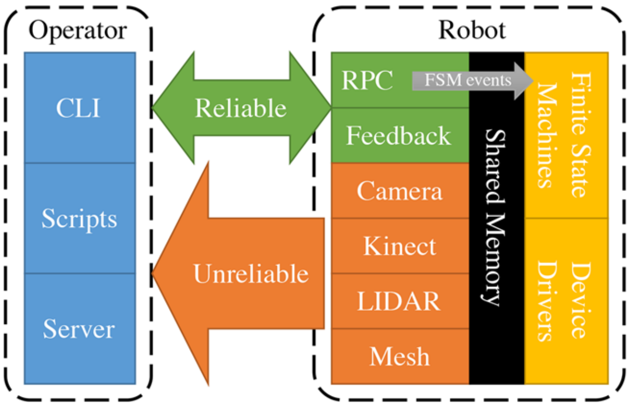


Figure 5. The software architecture centers around the reliability concerns of the DRC's network channels. Perception data ran through the unreliable channel, while commands and shared memory were synchronized over the reliable channel.

consumption of the AMD computer. Due to our minimalistic sensor setup and efficient motion-control code, a single computer was sufficient to handle all the motion and perception loads in real time. We use another NUC for the field computer that logs sensory data for later analysis.

3. SOFTWARE ARCHITECTURE

Our software framework has its roots in the RoboCup international robotic soccer competition (Yi et al., 2015). It provides a coherent way of organizing and executing processes for motion planning, sensor processing, autonomy, and communication interfaces. It is designed to be highly modular to support a variety of robotic hardware and be quickly ported on new humanoid platforms with minimal effort. Figure 5 depicts how processes are segmented by role and the way they interact with shared memory and the network channels.

3.1. Software Modules

To avoid a single point of failure, we maintain a number of separate processes that handle specific functions of the system. These processes can be restarted individually during the operation in case of failure. However, the interprocess communication can become a critical component. Since the robot uses a single computer with the Linux operating system, Unix domain sockets provide a viable transport mechanism for sending messages locally. Data messages are serialized using the MessagePack specification, and they can be directed either locally or remotely to other processes, a logging system, or a remote operator UI. Complementing the message passing system was a shared memory layout, where device drivers could read and write values, and configuration settings could be mutated on the fly. This shared memory approach is another advantage of a single computer design, where synchronization issues are avoided.

At the lowest level, there are a number of raw I/O processes. The motor control process publishes to four chains of the Dynamixel actuators at 120 Hz. The IMU process reads accelerometer, gyro, and filtered orientation data at 100 Hz. The camera process grabs camera frames from head and wrist cameras at 15 and 5 Hz, respectively. The auditory process monitors the microphone signal levels for volume spikes.

At the next level, we have various perception processes and upper- and lower-body motion controllers. Perception processes accumulate the sensory data, build a 3D mesh, and detect features before sending the results to the remote operator. Lower- and upper-body motion controllers receive high-level commands and generate motions for the lower or upper body to complete locomotion or manipulation tasks.

Finally, a number of finite state machines (FSMs) govern the high-level behavior of the robot. An overarching `BodyFSM` controls the underlying `MotionFSM`, `ArmFSM`, and `HeadFSM` modules. Each state machine is updated at 120 Hz to match the motor update rate. The `MotionFSM` handles the locomotion and balancing of the robot, the `ArmFSM` runs the upper-body control, and the `HeadFSM` controls head motions. The `BodyFSM` transitions among waypoint following, standing, and driving modes by sending signals to the other state machines. Once in a standing mode, the `ArmFSM` controls arm states, such as valve prepositioning, tucking arms, and entering teleoperation. Transitions are commanded remotely and forwarded via the remote procedure call system.

We tested our approaches using the Webots[1] simulator (Michel, 2004). The physical model in the simulation helped to identify torque limits for motions and validate dynamic motions before attempting on the physical robot. The simulated physics was updated at 8 ms, while the simulated sensors were updated at the same rates as the physical robot. Our operator systems could interact with the simulator or the real robot with minor configuration tweaks.

3.2. Communication Architecture

The DRC Finals allotted two network channels over which operators could communicate with the robot. A high-bandwidth channel allowed unidirectional packet flow from the robot to the operator with bandwidth around 300 megabits per second. The channel encountered significant blackout periods for "indoor" tasks, where all packets were dropped, in between 1 s bursts of no dropped packets. For "outdoor" tasks, no packets were dropped. A low-bandwidth channel operated at 9,600 bits per second bidirectionally, where 1,200 bytes per second could be

[1]Cyberbotics Ltd. Webots Commercial Mobile Robot Simulation Software. http://www.cyberbotics.com

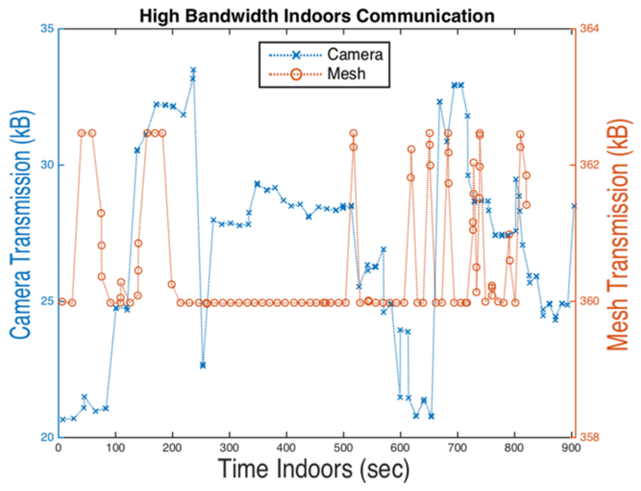


Figure 6. The network usage during 1 s high bandwidth openings showed that lossless mesh data were ten times larger than the high-resolution camera data.

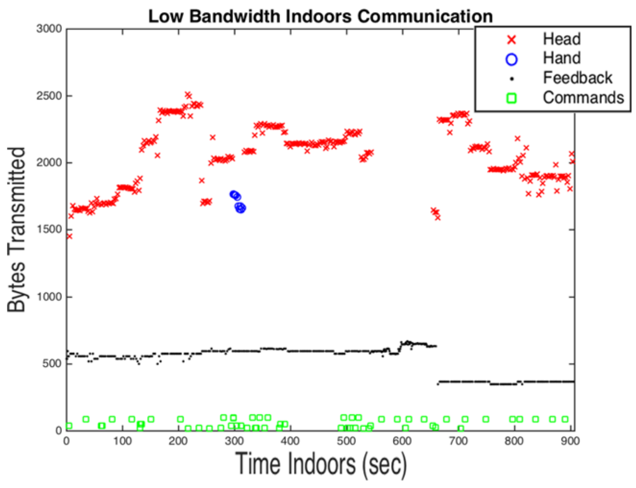


Figure 7. The upstream and downstream network usage over the low bandwidth link shows the predominance of the head camera information and the small size of the command packets.

transmitted from the robot to the operator, and another 1,200 bytes per second were allowed from the operator to the robot. This reliable channel would buffer, not drop, packets, so sending packets above the bandwidth limit would impact responsiveness.

3.2.1. High-bandwidth Architecture

We formed User Datagram Protocol (UDP) packets to transmit data from the robot to the operator over the high-bandwidth unidirectional channel, which requires no acknowledgment from the operator. We fragmented the UDP packets to 1,462 bytes in order to comply with the packet-filtering system, which allowed packets no larger than the maximum transmission unit (MTU) of 1,500 bytes. We uniquely tagged each UDP packet with a 10 byte preamble of information to reassemble the fragmented packets. With an 8 byte UDP header and a 20 byte IP header, we could transmit 1,462 bytes per fragment.

The high-bandwidth channel carried a cache of LIDAR returns to form a 2.5D (height map) mesh (Hebert et al., 2015) and camera frame streams. We chose to send compressed camera images at 15 Hz under outdoor conditions and 3 Hz under indoor conditions; we sent uncompressed mesh data at 1 Hz outdoors and 3 Hz indoors. Figure 6 shows the received data over time of this high-bandwidth data.

Since the 1 s window could not be predicted, a uniform 3 Hz attempted send rate allowed data to be sent during the unpredicted 1 s opening periods. Additionally, since the communication channel was implemented on a wireless network, the packets may be dropped due to physical link issues. We decided to burst send the same camera frame three times (at 3 Hz) in case any fragment was dropped. The burst data had the same preamble tags, so data could be assembled from any of the three bursts of data.

We failed to recover all three bursts of head camera packets to form a camera frame only once in 112 burst attempts when indoors in the second trial—we did recover two bursts that time. The average head camera frame was 27 kB. The mesh data were markedly different, and we did not use the burst mode. Since the amount of data was tremendously large as shown in Figure 6, with an average of 360 kB per chest LIDAR frame and 257 kB per head LIDAR frame, we sent the cache only at the 3 Hz rate. We recovered all three frames within the 1 s, opening only 24 of 59 attempts for the head LIDAR and 45 of 92 for the chest LIDAR for an aggregate rate of 45%. We recovered two frames 14 times for the head LIDAR and 11 times for the chest. Thus, the repetitive sending was tremendously valuable for large data so as to never miss the 1 s opportunity to send a frame of sensor information.

3.2.2. Low-bandwidth Architecture

The robot pose and nearby obstacles comprised the most important feedback that the operator needed at regular intervals. In case the robot came dangerously close to an obstacle, the operator could immediately issue a stop command. We encoded distance information that showed the nearest obstacles in a polar view. Additionally, we sent volume information as a single byte by processing the microphone stream.

Additionally, since regular camera feeds install much more confidence in the operator, we pushed to include camera images over the low-bandwidth link. By sending a JPEG image compressed in gray scale at a 160 by 90 resolution with a quality of 40, we could augment the pose

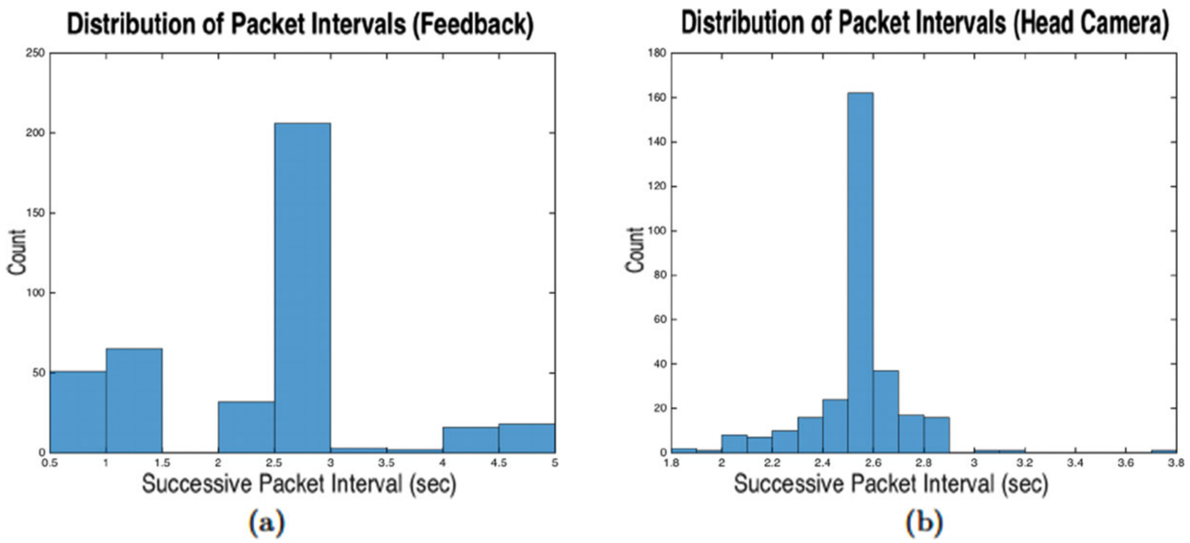


Figure 8. Feedback packets (a) were sent at three different rates, depending on how the cameras were enabled. When enabled, low bit-rate head image packets (b) arrived every 2.5 s.

information with a camera feed. Similarly, the wrist camera was compressed as a low resolution (80 by 60) color image, with quality 50. These reliable video feeds helped to save time aligning the hand with the valve and walking around in general.

Without camera images, the feedback was sent at 1 Hz, and with the head camera this became 0.35 Hz; feedback with the wrist camera was transmitted at 0.4 Hz. This feedback rate was modulated based on the actual size of the compressed image, which varied between frames. We did not wish to clog the channel, so the feedback would pause if we calculated that the low bit-rate channel was clogged. Shown in Figure 8 are histograms of the effective feedback interval for all feedback packets, and for feedback with low bit-rate head camera images in particular.

We utilized Transmission Control Protocol (TCP) packets on the low-bandwidth link to ensure that our commands were received by the robot since TCP implements packet resending and enforces order for proven reliability. Under the ZeroMQ[2] Request/Reply pattern, we commanded the robot with state machine events, target arm poses, and walk velocities in a remote procedure call (RPC) fashion. For safety, typical commands invoking shared memory or state machine events adhered to a structure defining valid memory segments that would not crash the robot. However, unsafe commands were allowed, but they were encoded as only the ASCII text of the command, executed as a protected Lua call to prevent crashes. Over the command channel, entire subsystems such as the motion, LIDAR, camera, or feedback could be stopped and restarted in case of failures. The bandwidth usage of typical remote shells vastly reduced the available channel bandwidth, so this process of restarting via RPC was a critical added feature. The bandwidth usage of the low-bandwidth channel, including sensor readings, state feedback, and commands, is shown in Figure 7.

3.3. User Interface

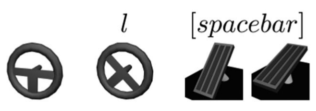
The user interface was structured as a hierarchy of control, shown on the operator side in Figure 5. At the lowest level, the operator interacts at the command line to issue commands, check the robot state, and observe sensor data. At the highest level, a graphical interface continuously displays sensor data and state feedback while allowing for mouse clicks and motion previews. In between are scripts that leverage models of the environment to execute primitive motions and display task-specific state information.

3.3.1. Command Line Interpreter

The RPC system exposes low-level access to the robot via a Lua-based command line interface (CLI). Here, the operator incurs a high cognitive load (Goodrich & Schultz, 2007), but it has the ability to inspect and modify joints, states, and configurations.

Scripted autonomy aided in the driving task, where the operators exclusively used the scripted motions for joint level motions. The chopstick-based wrist would turn the steering wheel by 45° increments, and the accelerator pedal was pressed in a similar fashion using the ankle pitch motor. In conjunction with the command line interpreter for head movements, the scripted autonomy reduced

[2]P. Hintjens. (2010) ZeroMQ: The Guide. http://zguide.zeromq.org/page:all



Console Output
```
Steering angle:  0
Steering angle:  45
2 sec burst
```

Figure 9. Virtual car parts provide a structured model of interaction.

cognitive load and increased reliability. Pressing "j" or "l," for instance, moves the steering wheel back and forth while simultaneously showing the user the angle of the wheel, while [$spacebar$] activates the accelerator pedal. This precluded the possibility of disastrous typos in the interpreter and reduced the lag between commands by eliminating typing in favor of hotkeys. The interactions with this interface is shown in Figure 9.

3.3.2. Graphical User Interface

Our GUI consists of a number of configurable HTML5 documents that visually show a number of perception cues utilizing both the low- and high-bandwidth channels. A typical setup of the graphical interface during indoor navigation, shown in Figure 10, uses one document for camera streams and one for a 3D viewport.

The low bit-rate gray-scale image is displayed alongside the high bit-rate camera image. On camera feed interfaces like this, the low bit-rate image continuously updates, while the high bit-rate image updates intermittently. An interface dedicated to only the high bit-rate image was used during outdoor network conditions when no blackouts were encountered.

The main 3D viewport shows the current state of the robot and the 2D obstacle indicator, which is updated using the low-bandwidth channel, the 3D mesh of the environment, which is updated using the high-bandwidth channel, and the target configuration of the robot, which is set by the operator. The robot state indication shows the current arm configuration and 3D pose of the robot in the reconstructed 3D mesh of the environment. The remote operator can move around the target configuration of the robot to make the robot walk to that position.

The low-bandwidth channel helps the operator to see the 3D pose of the robot and the 2D obstacles, which are color coded, where green rods are far away and red rods are nearby. A binning system is used to generate this obstacle information. Each bin contains the nearest LIDAR return within a certain field of view of the LIDAR (e.g., 30°–45° forms one bin, and 45°–60° forms another bin). These nearby points form a rough estimate of nearby obstacle information for the remote operator both for avoiding obstacles such as the door frame, and localizing with respect to nearby walls. The robot did not utilize this information for planning. When the robot is close to manipulation targets, the slowly updated 3D mesh can be used to finely guide the manipulation.

During manipulation, planned arm movements are previewed in the 3D viewport before the user allows execution; this is a similar strategy to that employed by other DRC teams (Johnson et al., 2015). Additionally, the interface shows how the robot will move in advance when the scripted motion is selected.

3.4. Perception

DARPA provided information about the competition environment ahead of time to the teams, but detecting and registering poses of items in the environment, such as the drill and valve, still proved challenging. However, much of the environment, from walls to steps and rough terrain, was planar, and registration of these planes was tractable and could provide useful cues for motion planning and operator awareness. For the manipulation tasks, the human operator used many perception information sources under the bandwidth restrictions, as automated grasping and manipulation proved complicated and untrustworthy.

3.4.1. Manipulation Vantage Points

Fine-grained manipulation tasks required many vantage points for successful completion within a reasonable time frame. Shown in Figure 11, the biggest source of confusion for the operator was depth perception. Since the view from the head could not give a sense of the distance to an object, and with LIDAR data yielding noise in distance measurements around 1 cm, more cues were needed.

The drill task in particular required fine-grained operator control. With the camera on the wrist of the robot, we could align the wrist to look at the drill. By cropping the image, we could even present the operator with a color image stream during the indoor network conditions. With color, fiducial markers such as colored zip ties and patterned tape become immensely useful in aligning the gripper to the trigger. With two perspectives, we reliably triggered the drill during practice sessions in reasonable amounts of time.

To confirm that the trigger was pressed, we measured the volume of ambient sound from the head and wrist via microphones coupled with the camera. This reduced the transmission from a sound stream into a single integer of the level. We measured the level before and after attempting to trigger in order to compare the relative sound levels, since a powered drill adds significant noise.

3.4.2. Plane Detection

Planar objects such as walls, floors, and stairs play an important role in understanding the environment (Nishiwaki, Chestnutt, & Kagami, 2012). In addition to localization cues, the relative pose from these planar objects influences stepping strategies for maximum stability and arm plans to avoid collisions. Our plane segmentation modules are designed to provide the geometry, i.e., normal, distance, and

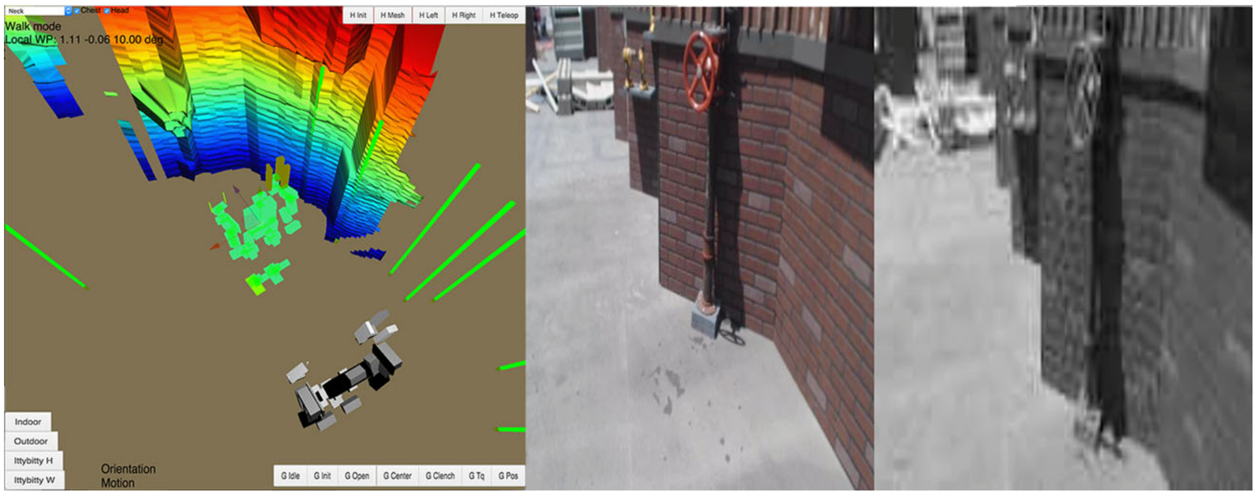


Figure 10. The user interface provides a variety of perception cues, which includes the current configuration of the robot (shown in gray rendering), the target configuration of the robot (shown in green rendering), 2D obstacles (shown in vertical rods), 3D mesh, and high and low bandwidth video feeds.

boundary points, of detected planes from LIDAR scans or colored depth images to facilitate the localization and locomotion of the robot.

We run a plane-detection algorithm on the onboard computer, which continuously estimates the pose with respect to walls. This helps the robot to approach a manipulation target and to avoid collisions in the environment. To identify multiple planes with arbitrary normals, clustering techniques have been commonly used (Chen, Taguchi, & Kamat, 2014; Holz, Holzer, Rusu, & Behnke, 2011). We chose to use the mean-shift clustering algorithm (Cheng, 1995) with an unknown number of clusters in the normal space. As the motion of the robot was constrained, the centers of previously found clusters were used as the seeds for the next frame. Connectivity in the projected 2D image was also considered so as to distinguish different instances of planes with similar normals. Figure 12 shows a plane segmentation result of the wall and the ground using LIDAR scans taken during the Finals.

We also use plane-detection algorithms on the operator side to find safe 3D foothold positions for the rough terrain, as shown in Figure 13. An automated region growing algorithm computes the centroid and inclination of each surface by optimizing a least-squares cost function. All the human operator needs to do is click on a desired foothold position on the surface.

4. MANIPULATION FRAMEWORK

Motion planning for the THOR-RD splits the upper-body and lower-body control, where upper-body manipulation routines are executed in a separate thread than lower-body locomotion control. The upper body includes seven degrees of freedom for each arm, as well as the waist yaw and pitch controls. The upper body alone represents 16 degrees of freedom (DOFs) available for any given manipulation task, implemented with independent position-controlled motors. The gripper functionality, utilizing current control, adds fine-grained control for grasping. In this section, we describe how we generated arm motions for redundant DOF arms using task-specific human preferences.

4.1. Optimizing Arm Plans with Human Preferences

For general purpose manipulation tasks, redundant DOF manipulators can avoid singularities and greatly enlarge the workspace (Hollerbach, 1985). However, resolving redundancy for the closed-form inverse kinematics (Chang, 1987) from the 6D arm end effector pose includes an optimization problem for free parameters (Shimizu, Kakuya, Yoon, Kitagaki, & Kosuge, 2008) that is hard to solve analytically.

Local Jacobian-based control (Siciliano, 1990) methods often encounter issues at singularities (Klein & Huang, 1983), and a number of attempts have been made to avoid this encumbrance (Buss & Kim, 2005; Chiaverini, 1997; Lloyd & Hayward, 2001; Nakamura & Hanafusa, 1986). Global planner approaches can avoid the singularity problems with local planners, but the global space can be intractably large. These planners include searching (Cohen, Chitta, & Likhachev, 2013) and random sampling (Stentz et al., 2015) to generate motions for high DOF systems, possibly augmented by a local Jacobian controller (Weghe, Ferguson, & Srinivasa, 2007). Finally, optimization planners can refine global plans, but often their seed trajectories limit the ability to avoid undesirable trajectories in a local minimum.

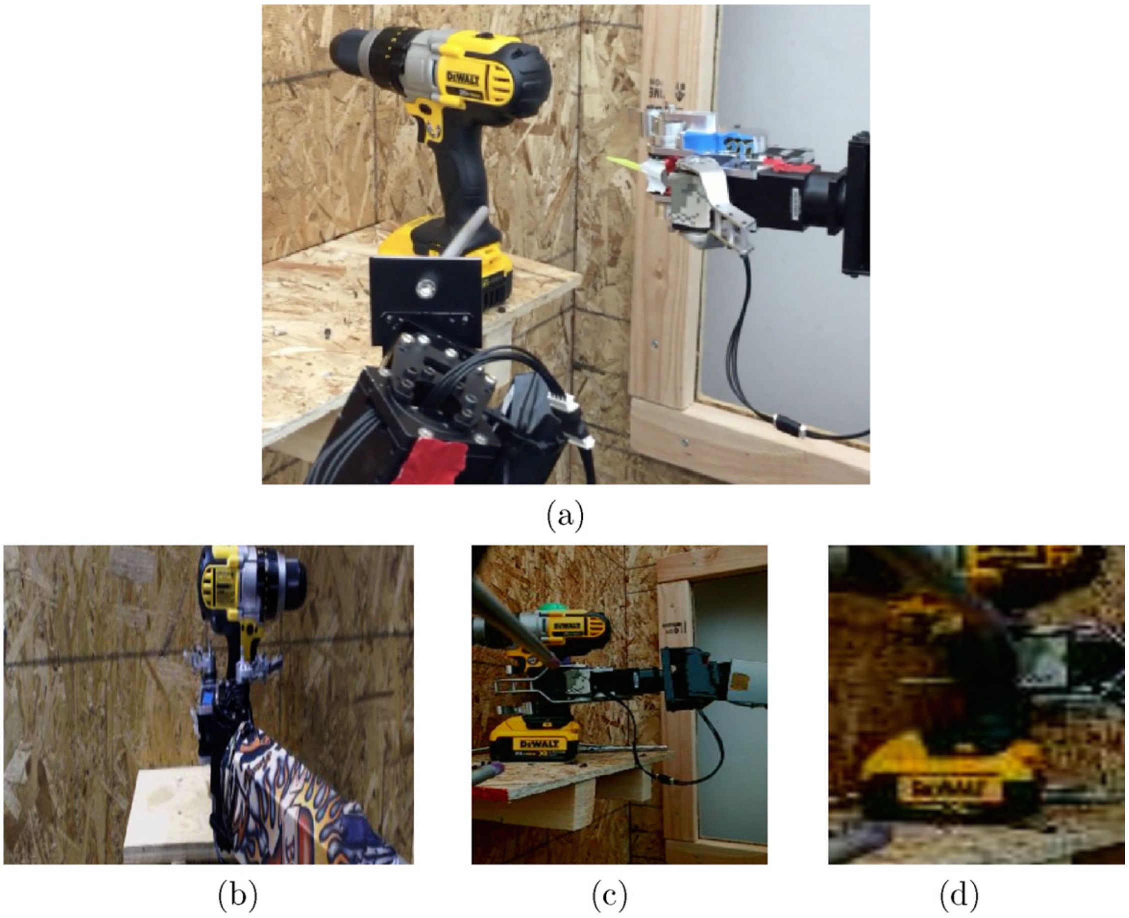


Figure 11. A secondary camera helps overcome the poor depth perception of the main camera. (a) Third person view. (b) The main camera feed. (c) Secondary camera feed on the high bandwidth channel. (d) Secondary camera feed on the low bandwidth channel.

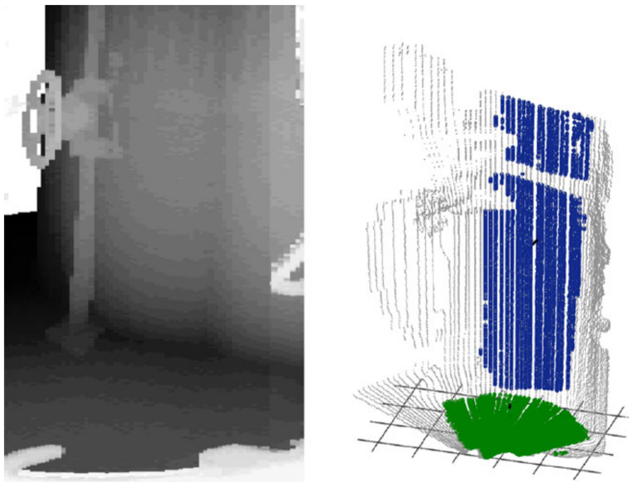


Figure 12. Automatic plane detection from LIDAR scans for localization.

These approaches can be problematic for long-term general purpose teleoperation usage in disaster response scenarios. Random search-based planners do not yield repeatable and predictable trajectories, thereby decreasing operator confidence. Additionally, the large search space for global planners can overburden the robot's onboard computer, which is constrained by both weight and power. An optimization-based planner using a single cost function will be suboptimal over a number of different tasks.

Our planner handles the redundancy of the manipulator, makes the resulting motion predictable and suitable for a given task, and lowers the complexity of the motion planning. By incorporating human operator input into the planning process for kinematic chains, high-level reasoning such as obstacle avoidance or self-collision checks can be embedded by following human intuition. Convex formulations of these costs become tractable for optimization, and initial trajectories are formed using an efficient Jacobian-based approach.

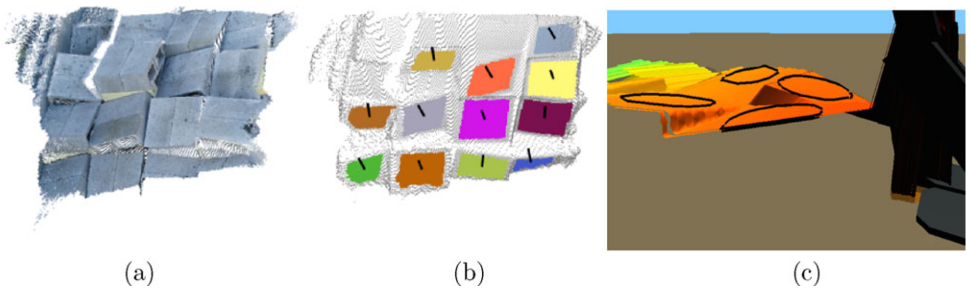


Figure 13. Semiautonomous selection of the footstep positions. (a) Original RGBD data. (b) Segmented planar surfaces. (c) Admissible foot landing positions selected by the operator.

Planning for high-dimensional robots does present a challenge in intuition for operators—complicated and unexpected motions may provide the optimal valid path. However, the operators are trained in many scenarios to understand the types of strange motions that can result, and the GUI presents a preview system before the robot executes a path. Additionally, the global optimization can undo restrictive constraints from the user during initial trajectory generation.

4.1.1. Task Space Motions

With a kinematic chain, motion paths $\mathbf{q}_{1:n_t}$ are planned in the space of joint angles in time, $\mathbb{R}^{n_t \times n_q}$, where n_t represents the number of discrete time steps and n_q represents the number of joints in the chain. For optimizing motion paths, we denote $\mathbf{q}_t$ to represent a set of joint angles on the kinematic chain at time step t.

Typically, the goal of motion planning is specified in task space, for instance the specifying final end effector transform. We introduce the variable $\mathbf{x} \in \mathbb{R}^{n_x}$ to define a task space coordinate, with $n_x < n_q$. As shown in Eq. (1), we define the first cost in the motion-planning optimization as the length of the path in task space, where smaller length paths are desired,

$$L_{\text{length}}(\mathbf{x}_{2:n_t}) = \sum_{t=1}^{n_t} \|\mathbf{x}_t - \mathbf{x}_{t-1}\|^2. \qquad (1)$$

Since we are generating a motion path in joint space, we must relate task space coordinates $\mathbf{q}_t$ to joint space coordinates via a Jacobian matrix, J_q, such that $J_q\dot{\mathbf{q}} = \dot{\mathbf{x}}$. With uniformly spaced time steps, Δt, $\dot{\mathbf{x}}_t = \mathbf{x}_t - \mathbf{x}_{t-1}$ and $\dot{\mathbf{q}}_t = \mathbf{q}_t - \mathbf{q}_{t-1}$ up to a constant scaling factor. The joint space reformulation is shown in Eq. (2),

$$L_{\text{length}}(\mathbf{q}_{2:n_t}) = \sum_{t=1}^{n_t} \left\| J_q(\mathbf{q}_t - \mathbf{q}_{t-1}) \right\|^2. \qquad (2)$$

Due to dynamic considerations of humanoids, jerky kinematic motions lead to disturbances that make the robot unstable. To avoid these jerky kinematic motions, we add a penalty for large accelerations in joint motion, shown in Eq. (3), where $\ddot{\mathbf{q}}_t \propto 2\mathbf{q}_t - \mathbf{q}_{t-1} - \mathbf{q}_{t+1}$ in discrete time,

$$L_{\text{acceleration}}(\mathbf{q}_{1:n_t}) = \sum_{t=2}^{n_t-1} \left|\left|2\mathbf{q}_t - \mathbf{q}_{t-1} - \mathbf{q}_{t+1}\right|\right|^2. \qquad (3)$$

Table I. Motion configuration preferences.

Similar Configuration	$\sum_t \|\|N_q(\mathbf{q}_t - \tilde{\mathbf{q}})\|\|^2$
Tucked Arm	$\sum_t \|\|N_q \mathbf{q}_t^{(2)}\|\|^2$
Range of Motion Use	$\sum_t \|\|N_q(\mathbf{q}_t - \mathbf{q}^{(m)})\|\|^2$

4.1.2. Human Preference Cost Functions

In typical motion-planning problems, additional costs are included to represent obstacle avoidance, self-collision, and other concerns. While object models can be used (Kohlbrecher et al., 2015), their path optimization costs increase complexity and incur computation penalties. Instead, we allow a human operator to use task-specific knowledge to inform simpler cost functions for joint configurations that do not affect the task space motion. The preferences are mathematically represented as simple convex functions, with easy to understand textual concepts for the operator. We must project them into the null space of the task Jacobian, $N_q = I - J^\dagger J$, to negate the effects on the task space, where $J^\dagger$ is the pseudo inverse of the rank-deficient task Jacobian.

As an example, in the DRC, the human preferences utilized l_2 norms, shown in Table I. The ith joint in the configuration is denoted $\mathbf{q}^{(i)}$, the middle of the range of motion is denoted $\mathbf{q}^{(m)}$, and $\tilde{\mathbf{q}}$ represents an "encouraged" configuration.

We are motivated to use these preference metrics based on real-world tests preparing for the Finals. In the door task, shown in Figure 14(a), we needed to keep the arm tucked away from the wall while still pushing the handle forward. In the drill task, the wrist motors can rapidly accumulate heat while holding the drill and eventually hit a thermal

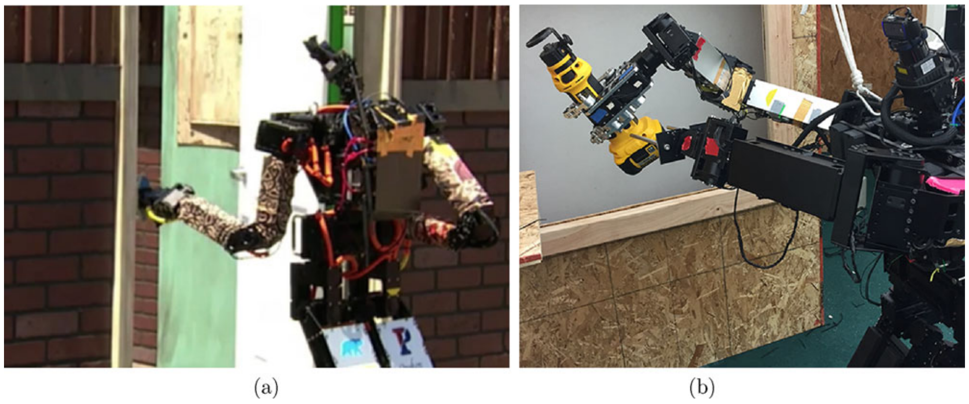


Figure 14. Two particular cases required human preferences for arm planning. (a) Tucked arm stances are required for obstacle laden situations, and (b) poor arm configurations can lead to thermal shutdown of actuators when holding heavy objects.

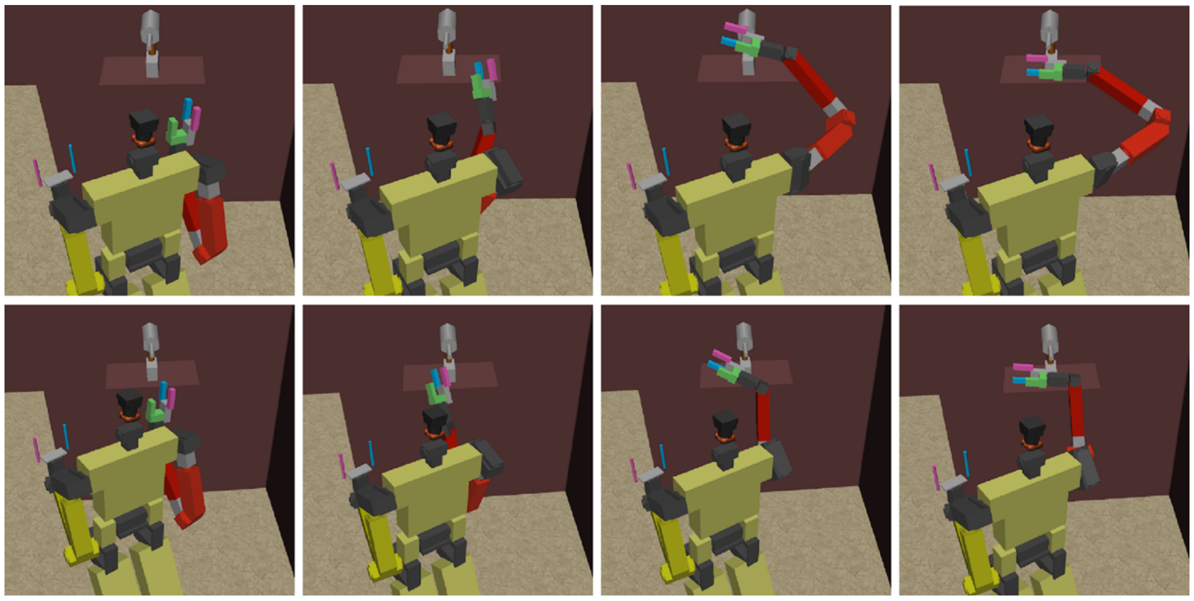


Figure 15. Above, the THOR-RD robot plans a path with the elbow protruding out with high manipulability in the middle of the joint range. Below, the robot plans a path with the elbow tucked in for maneuvering in tight spaces. The target transform for both paths is the same.

shutdown, shown in Figure 14(b), so we encourage configurations with the wrist motor axis aligned with the gravity vector. Figure 15 and 16 also show different arm trajectories according to the task specific human preferences.

4.1.3. Anytime Refinement

As J_q depends nonlinearly on $\mathbf{q}$, the optimization problem incorporating the previously described cost functions will not result in an efficient convex cost-function formulation. To mitigate this concern, we limit the values of $\mathbf{q}_t$ with respect to an initial, nonoptimal trajectory $\hat{\mathbf{q}}_{1:n_t}$. With the constraint in Eq. (4), we can formulate a convex optimization by approximating J_q as unchanging from the initial trajectory, thereby implementing a form of trust regions (Schulman et al., 2013),

$$||\mathbf{q}_t - \hat{\mathbf{q}}_t||^2 \leq \epsilon. \tag{4}$$

Using sequential convex optimization, we iterate the optimization many times, with the optimal solution set as the initial trajectory for the iteration, $\hat{\mathbf{q}}_{1:n_t} \leftarrow \mathbf{q}*_{1:n_t}$. In this way, we can continue refining the optimal trajectory until there is little change in the optimal cost. However, if we are content with even the initial plan, this iterative method can

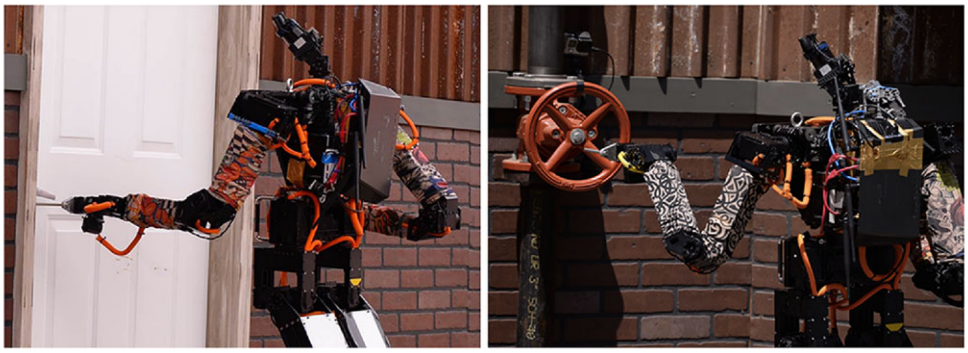


Figure 16. Two different arm preferences were utilized for the door (tight arm) and valve tasks (high range of motion) at the DRC Finals competition.

be thought of as an anytime planner, where the optimization is run until a time boundary is hit. During the DRC Finals, however, we found that the initial trajectory sent to the optimization program was refined enough to use in most cases.

4.2. Trajectory Generation

The calculation of the initial trajectory, $\hat{\mathbf{q}}_{1:n_t}$, will have a profound impact on the optimization solution time due to the sequential linearization. To form this trajectory, we use a Jacobian control law to drive the joint space motion, shown in Eq. (7). Filtering of the Jacobian controller includes clamps when out of the range of joint position and velocity limits,

$$J^{\dagger} = (\lambda I + J^T J)^{-1} J^T, \tag{5}$$

$$N = I - J^{\dagger} J, \tag{6}$$

$$\mathbf{q}_{t+1} \leftarrow \mathbf{q}_t + J^{\dagger}(\mathbf{x}_f - \mathbf{x}_t) + \alpha \cdot N(\mathbf{q}_t - \mathbf{q}_f). \tag{7}$$

We use a modified Jacobian pseudo inverse based on the selectively damped least-squares method (Buss & Kim, 2005) in order to avoid joint limits during motion (Na, Yang, & Jia, 2008). The pseudo inverse shown in Eq. (5), $J^{\dagger}$, yields a null space, N, that is full rank. This full rank null space will affect motion in the task space, but the degree of its effect can be tuned by α. Since N is positive-semidefinite, we can run a linear filter in time such that $N_{t+1} \leftarrow \beta N_t + (1-\beta)N_{t+1}$.

The controller update is run until the current task space pose is within δ, the final task space pose distance (metric M), or the trajectory exceeds a maximum number of time steps, $t_{\max}$, shown in Eqs. (8) and (9).

$$||\mathbf{x}_f - \mathbf{x}_t||_M \leq \delta, \tag{8}$$

$$t \leq t_{\max}. \tag{9}$$

The task space goal is specified as $\mathbf{x}_f$ and the initial joint configuration at $t = 0$ specified as $\mathbf{q}_0$. These specifications require no computation, but $\mathbf{x}_t$ requires forward kinematics computation at every time step, and $\mathbf{q}_f$ requires a one-time inverse kinematics solution.

4.2.1. Inverse Kinematics Optimization

The kinematic chain with redundant DOFs yields free parameters in the computation of the inverse kinematics for any task space goal, and thus the mapping from $f(\mathbf{x}_f) \rightarrow \mathbf{q}_f$ is not unique. To solve for the redundancy, we sample over the free parameters in order to optimize the same human preference costs, exemplified in Table I, of the path optimization. We constrain our path optimization so that the optimized path will respect this optimal inverse kinematics solution, $\mathbf{q}_f = \hat{\mathbf{q}}_f$, and the unchangeable initial configuration, $\mathbf{q}_1 = \hat{\mathbf{q}}_1$. The path is not defined, then, by a sequence of transforms, as used in other teleoperation systems (Zucker et al., 2015).

4.2.2. Joint Interpolation

If the trajectory exceeds $t_{\max}$ before approaching the task space pose within δ, then a secondary controller must drive $\mathbf{q}_{t_{\max}}$ to $\mathbf{q}_f$. In this case, we use a simple joint interpolation procedure to linearly drive the joint configuration to $\mathbf{q}_f$, shown in Eq. (10), with the stop condition in Eq. (11),

$$\mathbf{q}_{t+1} \leftarrow \mathbf{q}_t + \Delta q, \tag{10}$$

$$\Delta q \geq q_f - \mathbf{q}_t. \tag{11}$$

The trajectory generated from thejoint interpolation method will cause much higher costs in the task space length than the Jacobian controller. If the Jacobian controller is used exclusively, then the trajectory is very close to the shortest path in task space after the initial configuration. Thus, only the null space motion would need optimization; however, there is a tradeoff in task space length and null space motion for meeting human preference.

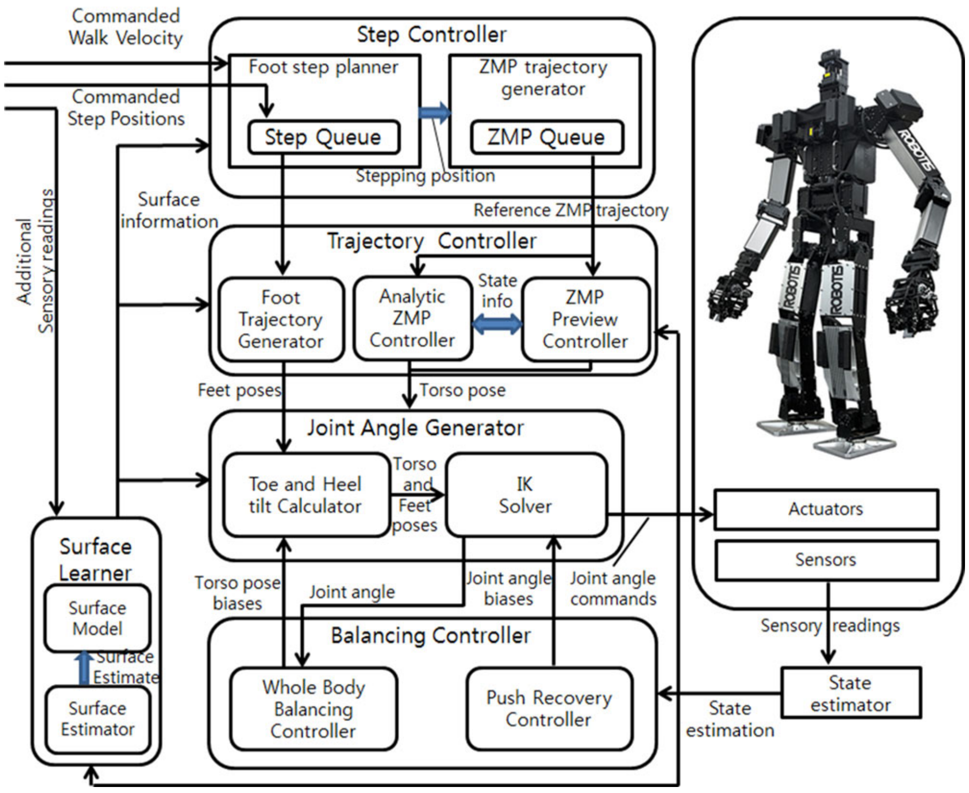


Figure 17. The structure of the locomotion and balancing controller provide a hierarchy where the surface inclination informs the high-level step controller and the low-level joint angle generator.

5. LOCOMOTION AND BALANCE CONTROL

For humanoid robots, moving quickly and reliably in unstructured environments remains a challenging problem. The DRC Finals in particular require more agility, stability, and versatility from the robot than even the DRC Trials. The robot must progress through the test environment and complete a number of tasks sequentially. Unlike the DRC Trials, where each task has a separate time limit with nearby starting positions, the Finals dictate a short, hour-long time limit with restart positioning only for special cases. The lack of safety measures—no belay—means that a single fall may catastrophically ruin the whole trial. Finally, the robot should be able to traverse uneven terrain without precise prior mapping and using limited remote control. In this section, we describe how we designed our locomotion and balance controller to fulfill those requirements.

5.1. Walk Controller Structure

Our locomotion and balancing controller consists of a step controller, a trajectory controller, a joint angle generator, and a balancing controller, as shown in Figure 17. The step controller receives a control input and generates the next step position with the corresponding ZMP trajectory. There are three control modes: the *direct* mode, which controls the locomotion by specifying the target velocity; the *target* mode, which autonomously generates optimal step positions from the current perceived robot pose to reach the given target pose; and the *special* mode, which is controlled by specifying the landing position for the next step. The trajectory controller generates the foot and torso trajectory, which uses a hybrid locomotion controller (Yi, Hong, & Lee, 2013) to switch dynamically between a standard ZMP preview controller that uses linear quadratic optimization and a reactive ZMP-based controller that uses a closed-form solution of the linear inverted pendulum equation.

The main benefit of this approach is that it provides both latency-free control of the next step position and a generalized step motion that is required for more challenging terrains. Utilizing these two controllers, our locomotion and balancing controller reacted quickly to unknown terrain patches.

The joint angle generator calculates the desired joint angles of the robot, tilting the feet around the toe or heel for improved performance, and finally the balancing controller stabilizes the robot using a number of stabilizing strategies.

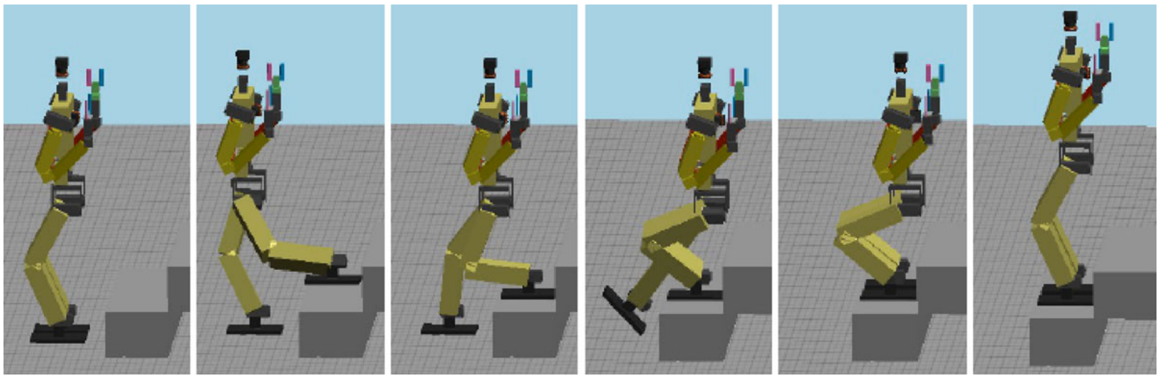


Figure 18. The THOR-RD robot can climb the staircase in simulation. The toe and heel lift controller was necessary to maximize the usability of the leg kinematics.

5.2. Walk Posture

As the THOR-RD platform has fairly short leg dimensions, it is not feasible to climb high steps using a quasistationary walk due to knee and shin strike problems. For the DRC Trials, we used a shortened foot with fast dynamic stepping for the rough terrain task. However, reducing the foot size negatively affected the overall stability of the robot. Fast dynamic stepping requires precise knowledge of the surface geometry and well-calibrated joint actuators; this approach does not work well in uncontrolled environments.

A solution for the knee and shin strike problem that occurs when climbing up steep staircases is to use a birdlike walk posture where the knee bends in the opposite direction. Additionally, with the knee behind the robot, better surface perception is afforded since the knee is no longer blocking the view of the head camera or chest lidar. On the other hand, this posture will lead to shin and knee strike problems when climbing down steps, which can happen at the later stage of the uneven terrain. We have found that this is not a major problem, as we can find a foot step plan that is clear of shin strike for most cases, with help of the heel and toe lift control.

Like most humanoid robots, the THOR-RD platform has an asymmetric knee design that can only bend to one direction, so the whole leg must face backward to use the birdlike walk posture. With THOR-RD's waist yaw joint, we rotate the upper body a full 180°; the robot can change the walk posture actively during the run. Initially, we planned to change the posture only for the terrain traversal tasks, but we decided to use the posture for all tasks since the birdlike knee posture provides counterbalancing advantages for manipulation tasks, and using a single posture simplifies turning the walk parameters.

5.3. Heel and Toe Tilt Controller

Fast dynamic stepping was used in the DRC Trials with the THOR-OP platform since its knee torque is too little for slow, quasistationary stepping. The new THOR-RD platform has higher structural rigidity and joint torques to reliably perform quasistationary walking. It still has limited leg dimension compared to the step heights it must climb. To overcome this kinematic constraint, we utilize an automatic tilting of the foot around the toe or heel edges to stay in the stable double support phase longer. Also, the birdlike walk posture tends to cause the robot to land on its toes due to knee flex, and this makes the robot unstable while walking fast forward. We use the heel and toe tilt controller to ensure that the robot strikes the ground with its heel first. Figure 18 shows the THOR-RD robot climbing the staircase utilizing heel and toe tilt motions.

5.3.1. Trajectory Calculation

To support the heel-strike, toe-off walking gait on flat surfaces, we extend our piecewise linear ZMP trajectory (Yi et al., 2013) so that the ZMP moves linearly from heel to toe in a single support. For slow walking over uneven terrain, instead of moving the ZMP position, we fix the ZMP position to the center of the current support polygon to maximize stability. Once the reference ZMP and foot trajectories are generated, we use our hybrid walk controller to generate the torso trajectory, which uses both a ZMP preview controller and an analytic ZMP controller.

5.3.2. Automatic Toe and Heel Lift Angle Calculation

Toe or heel lift is needed when the leg length is not long enough for the reference torso and foot poses. We denote the projected ankle position over the landing surface by x_F, and the surface roll and pitch angles as ϕ_s and θ_s, as shown in Figure 19. The target transform on a surface with zero tilt angle is then

$$Tr_F = T(x_F)R_y(\theta_s)R_x(\phi_s), \tag{12}$$

where T is the translation matrix, and R_x and R_y are rotation matrices around the axes x and y. With the ankle height,

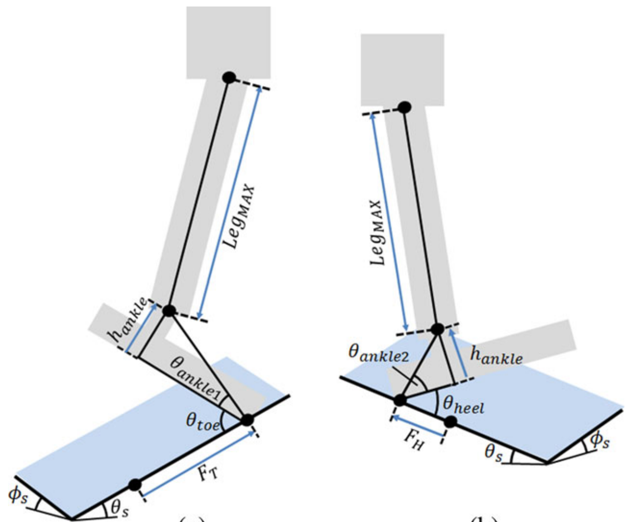


Figure 19. The toe and heel tilt were used to adapt to the inclined surface of the DRC Finals. (a) Heel lift. (b) Toe lift.

h_{ankle}, the position of the ankle is

$$x_{\text{ankle}} = Tr_F T([0, 0, h_{\text{ankle}}])[0, 0, 0, 1]^T. \tag{13}$$

We have the following kinematic constraint:

$$\|x_{\text{ankle}} - x_{\text{hip}}\| \le \text{Leg}_{\text{max}}. \tag{14}$$

In case Eq. (14) does not hold, we can lift the heel by θ_{heel} around the toe contact position to increase the effective length of the leg. The leads to the new ankle transform,

$$Tr_{\text{ankle}H} = Tr_F T([F_T, 0, 0])R_y(\theta_{\text{heel}})T([-F_T, 0, h_{\text{ankle}}]), \tag{15}$$

where F_T is the distance between the projected ankle axis to the toe contact position. Likewise, we can lift the toe by θ_{toe} around the heel contact position, resulting in the ankle transform

$$Tr_{\text{ankle}T} = Tr_F T([-F_H, 0, 0])R_y(-\theta_{\text{toe}})T([F_H, 0, h_{\text{ankle}}]). \tag{16}$$

We can solve the following equations to calculate the minimum toe and heel lift angles that satisfy the kinematic constraints:

$$\|Tr_{\text{ankle}T}[0, 0, 0, 1]^T - x_{\text{hip}}\| = \text{Leg}_{\text{max}}, \tag{17}$$

$$\|Tr_{\text{ankle}H}[0, 0, 0, 1]^T - x_{\text{hip}}\| = \text{Leg}_{\text{max}}, \tag{18}$$

which have a simple closed-form solution with zero surface inclination angles. For the general case, we use a newton solver to find solutions numerically. We use the convergence threshold of 0.5°, which causes the newton solver to terminate in fewer than 10 iterations for all the cases we have tested.

Once the lift angles are found, the type and amount of lift angle are determined based on the foot configuration and the range of motion of the ankle joint. In addition, the minimum lift angle can be manually specified even if the lift is not needed to generate the toe-off, heel-strike walk motion.

5.3.3. Joint Angle Calculation

Once the foot tilt angle is determined, we use inverse kinematics to generate the leg joint angles. The THOR-RD robot we use has three hip joints intersecting at a point and zero knee offsets, so the following relation holds:

$$\begin{aligned} Tr_{\text{ankle}} = {} & Tr_{\text{hip}} R_z(q_0)R_x(q_1)R_y(q_2) \\ & \times T([0, 0, -d_{\text{ULEG}}])R_y(q_3) \\ & \times T([0, 0, -d_{\text{LLEG}}])R_y(q_4)R_x(q_5), \end{aligned} \tag{19}$$

where d_{ULEG} and d_{LLEG} are upper and lower leg link lengths, and q_i are joint angles. We can first calculate the knee angle q_3 by rearranging Eq. (19),

$$\begin{aligned} Tr_{\text{hip}}^{-1} Tr_{\text{ankle}} = {} & R_z(q_0)R_x(q_1)R_y(q_2) \\ & \times T([0, 0, -d_{\text{ULEG}}])R_y(q_3) \\ & \times T([0, 0, -d_{\text{LLEG}}])R_y(q_4)R_x(q_5). \end{aligned} \tag{20}$$

If we denote $Tr_{\text{LEG}} \equiv Tr_{\text{hip}}^{-1} Tr_{\text{ankle}}$, then

$$\begin{aligned} ||Tr_{\text{LEG}}[0, 0, 0, 1]^T||^2 = {} & d_{\text{ULEG}}^2 + d_{\text{LLEG}}^2 \\ & -2 d_{\text{ULEG}} d_{\text{LLEG}} \cos(q_3), \end{aligned} \tag{21}$$

$$q_3 = \arccos\left(\frac{d_{\text{ULEG}}^2 + d_{\text{LLEG}}^2 - ||Tr_{\text{LEG}}[0, 0, 0, 1]^T||^2}{2 d_{\text{ULEG}} d_{\text{LLEG}}}\right), \tag{22}$$

and we can calculate the ankle angles q_4 and q_5 by rearranging Eq. (19) as

$$\begin{aligned} Tr_{\text{LEG}}^{-1} = {} & R_x(-q_5)R_y(-q_4)T([0, 0, d_{\text{LLEG}}]) \\ & \times R_y(-q_3)T([0, 0, d_{\text{ULEG}}]) \\ & \times R_y(-q_2)R_x(-q_1)R_z(-q_0). \end{aligned} \tag{23}$$

If we denote the following:

$$Tr_{\text{ILEG}} \equiv Tr_{\text{LEG}}^{-1} \tag{24}$$

$$M \equiv T([0, 0, d_{\text{LLEG}}])R_y(-q_3)T([0, 0, d_{\text{ULEG}}]), \tag{25}$$

then we have the following relationship:

$$Tr_{\text{ILEG}}[0, 0, 0, 1]^T = R_x(-q_5)R_y(-q_4)M[0, 0, 0, 1]^T, \tag{26}$$

which will lead to the following equations:

$$Tr_{\mathrm{ILEG}}\{0,3\} = M\{0,3\}\cos(q_4) - M\{2,3\}\sin(q_4), \quad (27)$$

$$\begin{aligned} Tr_{\mathrm{ILEG}}\{1,3\} = {} & M\{0,3\}\sin(q_4)\sin(q_5) \\ & + M\{1,3\}\cos(q_5) \quad (28) \\ & + M\{2,3\}\cos(q_4)\sin(q_5), \end{aligned}$$

which can be rewritten as a second-order equation of $\sin(q_4)$ and $\sin(q_5)$ and solved in a closed form. Finally, we rearrange Eq. (19) as

$$R_z(q_0)R_x(q_1)R_y(q_2) = Tr_{\mathrm{LEG}}R_x(-q_5)R_y(-q_4)M. \quad (29)$$

If we denote $H \equiv Tr_{\mathrm{LEG}}R_x(-q_5)R_y(-q_4)M$, we can obtain

$$q_0 = \mathrm{atan2}(-t\{0,1\}, t\{1,1\}), \quad (30)$$

$$q_1 = \arcsin(-t\{2,1\}), \quad (31)$$

$$q_2 = \mathrm{atan2}(-t\{2,0\}, t\{2,2\}). \quad (32)$$

5.4. Balancing Controller

Unlike the linear inverted pendulum model (LIPM) we use for dynamics calculation, the physical robot has distributed mass, which can seriously affect the stability of the robot if not correctly accounted for. The robot has to withstand external perturbation from various sources, which includes uneven terrain, contact with the environment, reaction force from manipulating an object, joint position error, and structural flex. In particular, the competition environment at the DRC Finals was built upon a surface with a global inclination severe enough to make many humanoid robots fall down. In the following subsections, we describe our approach to stabilize the robot.

5.4.1. Full-body Balancing

When humanoid robots are used for manipulation tasks, the difference arm configuration affects the overall center-of-mass location, and the robot must compensate for it to keep balanced. For the DRC Trials, we calculated the upper-body COM location based on current arm configuration, and we shifted the torso position so that the upper-body COM remained fixed. Leg masses are not considered, as we assume that all manipulation takes place when the robot is standing still with the preset leg stance. As the THOR-OP robot has a fairly heavy torso and lightweight legs, this approach has worked well in practice.

However, the new THOR-RD robot has quite a different mass distribution compared to the THOR-OP robot. Leg masses have increased significantly due to the increased actuator capacities and repositioning of the batteries, and the torso and hip have become significantly lighter.

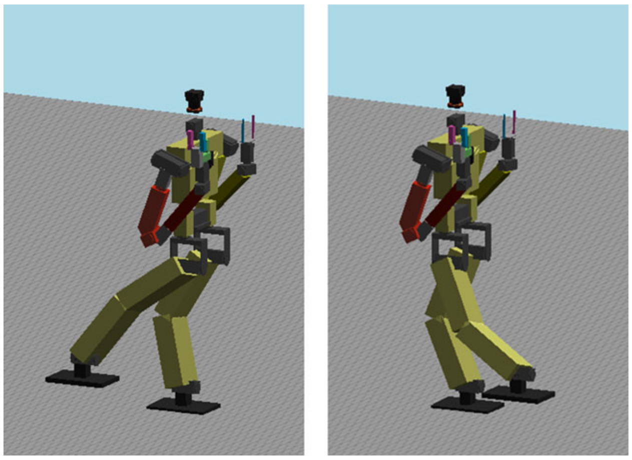


Figure 20. The whole-body balancing controller modulates the horizontal torso position to stay balanced.

Thus, now we use a fine-grained mass model to calculate the COM location of the whole robot, and iteratively update the torso position to compensate for the COM error. The balancing controller is now always active, using the reference COM position from the walk controller instead of the middle point of two foot positions as the target COM position. As a result, the robot can now correctly balance in the slow single support phase while performing arm motions. Figure 20 shows how the robot shifts torso position to remain balanced. We validated the full-body balancing controller using the THOR-RD robot with onboard force torque sensors, and we found that the measured COM error is less than a centimeter in the worst case.

5.4.2. Global Surface Adaptation

Most bipedal locomotion controllers assume a flat surface to walk on, and any unevenness of the surface can seriously hamper the stability of the walking robot. Unfortunately, the testing environment for the DRC Finals had a fair amount of global inclination, and we found that the amount of lateral inclination can induce landing timing errors, leading to a fall in the worst case.

We use a simple approach to handle this problem. We assume that the surface has a uniform global inclination, and we generate the walk motion based on it. As the actual surface gradient is not uniform, we use IMU feedback to adapt to a new surface gradient when the robot is not moving. To prevent the robot from walking into a region with a very different surface inclination, we limit the maximum amount of distance the robot can move at a time. Although simple, this approach has worked well in practice and has negligible time loss as the low communication bandwidth allowed the robot enough time to adapt to the new surface.

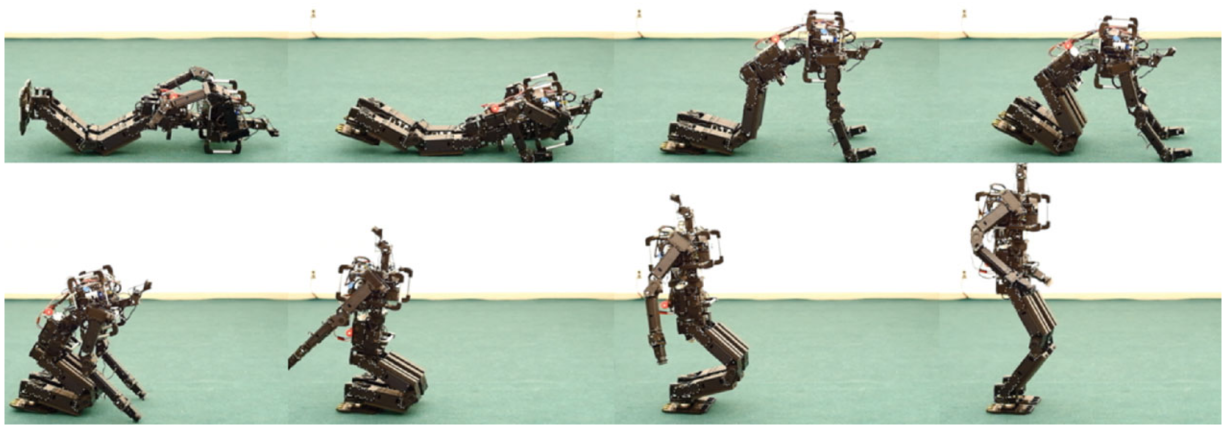


Figure 21. The THOR-RD robot performs its getup motion from a prone position. This behavior works reliably when the robot is facing down, and research into getting up from other postures is underway.

5.4.3. Push Recovery Controller

In the uncontrolled environment, the robot will confront many sources of perturbation that can destabilize the robot, such as contact with the environment, uneven surfaces, or joint tracking errors. For the DRC Trials, we used the ankle strategy, which modulates the ankle joint target position to counter the perturbation, and the stop strategy, which stops the robot in the most stable double support stance when the detected perturbation is large.

For the DRC Finals, we also utilized an adaptive landing timing controller and step strategy to handle lateral disturbances thanks to the reactiveness of the analytic ZMP walk controller. We used the IMU and joint encoder measurements to determine whether the following foot landing is too early or too late, and we adjusted the landing timing and landing position of the current swing foot. We tested the push recovery controller extensively, including having the robot walk over various irregular terrains and kicking the robot while walking in place.

5.4.4. Self-righting Controller

One of the main differences between the DRC Trials and the DRC Finals is that robots are deployed without a gantry system to protect it from falls. A humanoid should have the ability to get back on its feet after a fall, and this ability was included for DRC Finals qualification. We made a simple keyframe-based whole-body motion that can reliably make the robot stand up from a prone position, shown in Figure 21. However, as a fixed motion only works in limited circumstances, a more general approach is required to handle different fallen body configurations and surface geometries.

We have developed a machine-learning-based algorithm to generate self-righting motions from different body configurations (Jeong & Lee, 2016). The approach is validated with experiments using a scaled-down DARwIn-OP humanoid robot. One big challenge in applying machine-learning methods to humanoid robotics is the high-dimensional and continuous state and action spaces. To solve this problem, we discretized the state space, including all joints and body attitude information, using an expectation-maximization algorithm to cluster various poses.

To learn a path in the state state, we applied a Q-learning algorithm to repeated trials in a simulated environment. This approach effectively generated self-righting motions from a wide range of fallen body configurations, and even performed better than previous keyframe-based motions. The algorithm can be generalized to other humanoid robots, by means of using robot-specific kinematic subroutines. Unfortunately, we did not attempt this behavior in the competition due to concerns about damage after a fall. Self-righting attempts with damaged hardware can be futile and further damage the robot. In the future, we would like to provide the means to identify structural damage.

6. RESULTS

While the THOR-RD system was being completed, we tested on the THOR-OP platform from the Trials with modified arms and prototype grippers. End effectors and arm lengths were validated in simulation as well. Our testing included the Maxwell Pro network shaper, which helped to validate our network usage strategy.

We skipped the Egress task because the risk of a fall was high, which would compromise an entire run. Before attempting on the physical robot, all the other tasks were completed in a simulated environment when tried separately. With the physical robot, we did not test the Rough

Terrain or Debris tasks, and we eventually decided to bypass that section.

At the DRC Finals competition, Team THOR completed the Driving, Door, and Valve tasks for both runs. We used the same test field for both days, but unfortunately the severe surface irregularities on that field from patched asphalt caused the robot to fall down between the Valve and Drill tasks for both runs. On the first day we scored three points in 48 min, which includes the 10 min penalty for manual egress. On the second day, we scored the third point in a much quicker 28 min, including the penalty.

6.1. Driving

We use a periodic stop-and-go approach that was successfully followed in the DRC Trials to drive the Polaris vehicle. Although this method is slower than continuous driving with a dynamic model of the vehicle, it works fine with minimal prior testing. Steering is accomplished using the wrist yaw actuator that lies coaxially with the steering wheel column. Frontal movement is controlled by a timed pedalling motion, where the engine braking stops the vehicle when the pedal is released.

Testing before the competition revealed that the difficult part of the driving task is not the driving itself but setting up the robot in the seat and taking the robot out of the car reliably. There was no difficulty setting up the THOR-OP robot for driving during the Trials. However, with THOR-RD, we found that the slimmed down torso and shorter reach of the steering arm requires more precise positioning of the robot. Furthermore, the birdlike walk posture for locomotion complicates fitting the robot inside the cockpit due to the limited range of the knee joint.

To solve these problems, we placed the robot facing backward in the seat and used one arm to support the upper body. When the mounting of the robot is completed, the head rotates 180° to provide a frontal video feed while driving. We successfully completed the task for both runs, where the second run was completed significantly faster than the first one. Figure 22 shows the THOR-RD robot driving the Polaris vehicle during the competition.

6.2. Egress

During the early stages of development, we brainstormed possible ways to complete the egress task with the THOR-RD platform. One method included putting the robot sideways in the cockpit with both feet in the air. A cable system would control the gas pedal using the gripper hand. However, we decided to skip the Egress task because we were not sure of the reliability of our position-controlled actuators in multiple contact situations where the thermal shutdown of any actuator would ruin an entire run. Instead, we designed a motion sequence that partially turns off actuators to help field operators take out the robot.

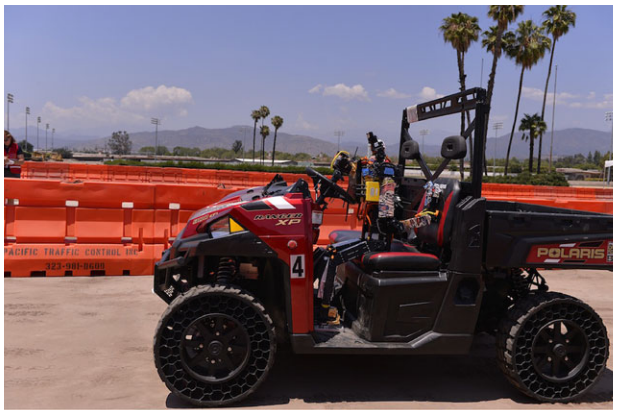


Figure 22. THOR-RD drives the vehicle with its head rotated 180°. Due to the default birdwalk knee configuration, the robot is mounted backward in the car.

6.3. Door

In the Trials, we found the Door task to be one of the harder manipulation tasks, as the planning of a long pull motion was not trivial. The door kept closing due to strong winds. Walking while pressing the loaded door made the locomotion unstable and inconsistent. For these reasons, we chose to approach the door sideways and cross the door frame by sidestepping. In this way, there is more margin for error, and the robot is more robust to lateral perturbations than frontal ones.

The DRC Finals Door task has been largely simplified, presenting one push type door that swings open fully once the latch is unlocked. However, as the robot has to complete all tasks in sequence, speeding up the task becomes a high priority. For these reasons, we decided to pass the doorway by walking forward, which is much faster than sidestepping but has a high chance of collision.

We found that to open a push type door while facing the door, the robot should align closer to the door in order to push the door ajar far enough. We employed the preference-based arm motion planner to generate the arm ready motion for tight spaces. The door-opening motion included optional waist rotations to leverage a larger end effector workspace for windy situations such as the DRC Trials. Making the robot go straight through the door frame is not a trivial task and requires good positioning and situational awareness. We use the head LIDAR to guide the robot to the center of the nearby door frame, as shown in Figure 23. The standard walk motion can move the robot forward without touching the frame. If the robot contacts the door frame, the adaptive landing timing controller and push recovery controllers help the robot to remain standing.

On the first trial, we needed almost 6 min to open the door, as the handle was verified to be broken and would

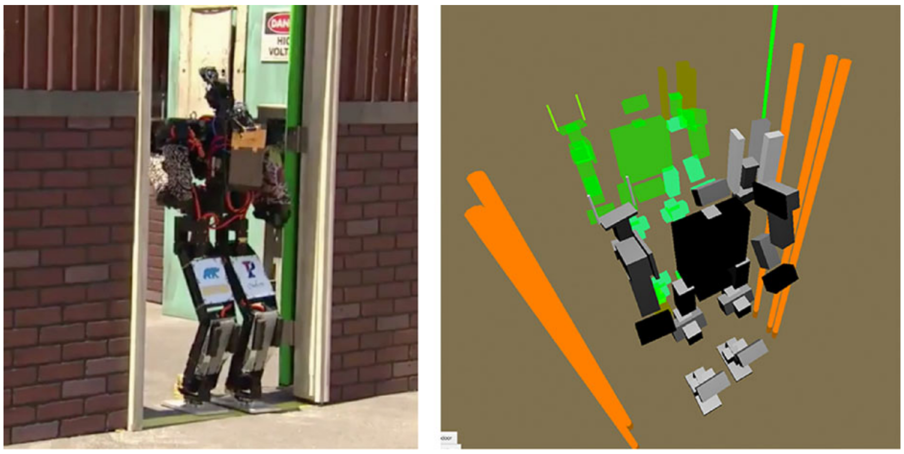


Figure 23. The operator guides THOR-RD through the doorway with the help of LIDAR feedback over the low bandwidth channel.

open the door when turned down but not up. It took an additional 6 min to cross the threshold, as we poorly positioned the robot in the door frame, and the robot came in contact with the frame while walking. However, thanks to our safeguards, the robot kept walking forward and completed the task. On the second trial, it took 1.5 min to swing open the working door, and an additional 1.5 min to pass the threshold into poor network conditions.

6.4. Valve

For both DRC competitions, we exploit the continuous rotating wrist yaw joint and robust passive hand to complete the task quickly. The passive hand, which consists of two parallel rods, proved to be very robust to minor alignment errors and helped us to receive the best in task award for Valve at the Trials.

For the DRC Finals, we reduced the length of the passive hand, as it must operate in other tasks as well. This reduced the margin of error for positioning the robot. The approach distance significantly increased due to the sequential nature of the tasks. Thus, the main focus during preparation became approaching. We identified good stance offsets from a variety of valve positions that led to fast times to engage the valve with the gripper.

However, at the competition, the sloped and unpredictable terrain meant that the walking engine could not be trusted to make the fine grained steps to the best stance. We decided to stop walking at much further distances to the optimal pose than planned. The preference-based planner, set to occupy the middle of the joint’s range of motion, found smooth trajectories in this unexpected arm workspace.

On the first trial, we took 7 min to turn the valve, needing to correct the alignment of the arm after inserting the chopsticks inside the valve handle. On the second day, we aligned well on the first attempt and took just under 5 min to complete the task. Figure 24 shows the robot lining up its arm to the valve. While engaging and disengaging the valve was executed with the planner, the valve turning motion was conducted by direct joint angle control of the wrist yaw joint, while being observed by the operator using the low bandwidth channel.

6.5. Drill

Due to the torque limit of wrist actuators, we chose to use the lighter gun-type drill for both of the DRC competitions. In the Trials, we used a two-finger gripper with a fixed palm that requires a precise alignment of the gripper to successfully trigger the drill. Although we successfully picked up and triggered the drill, it took a considerable amount of time, which is not desirable for the more time-limited DRC Finals.

We focused on iterating on the gripper design in order to grab and trigger the drill robustly with some amount of alignment error. In testing, we used frequent image updates on the low-bandwidth channel to provide low latency situational awareness. The image updates included the wrist camera feed, so both arms were in motion during the task. We fine-tuned the wrist camera using joint level motions, while the actuated gripper was moved into place using the preference-based planner. One issue found from testing is that the wrist roll actuator can reach thermal shutdown with some arm configurations. To mitigate this situation, we set the planner preference to avoid such configurations.

We found that on the new gripper with the actuated thumb finger, grabbing and triggering the drill is considerably easier. We could consistently grab and trigger the drill through remote operation. After fine-tuning the pregrasp pose of the gripper, we command the gripper fingers individually using current control. Before and after command-

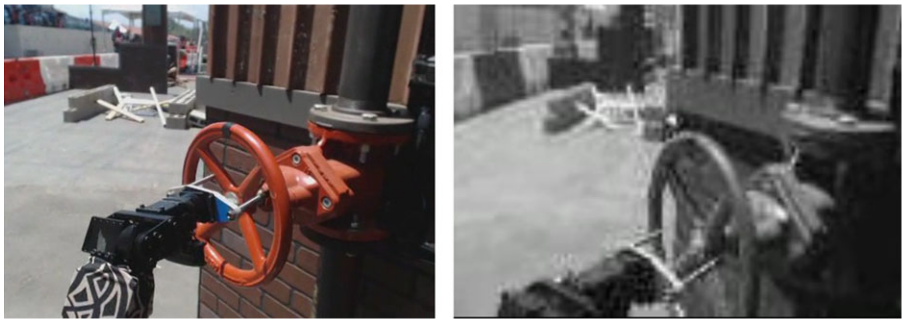


Figure 24. The high and low bandwidth view during the DRC Finals Valve task: The robust passive hand allows for quick alignment, and the low-resolution image feed provides immediate feedback during operation.

ing the fingers, we request volume levels from the robot's microphones. This provided a reliable way to ascertain the trigger state, as an activated drill saturated the microphone. Shown in Figure 11 are side-by-side color and gray-scale images that provided cues for depth perception and trigger alignment.

Unfortunately, we did not progress to the Drill task during the actual competition, as the robot fell down in both runs after stepping on the surface irregularities in front of the Drill task setup, shown in Figure 30.

6.6. Surprise

The surprise task was chosen randomly for each day of the Finals from a number of manipulation tasks. The potential set of tasks was revealed in advance of the competition.

We first set up simulated models of those tasks to check feasibility within the workspace of the THOR-RD robot. We then set up mockups of the task targets to test with the physical robot. We found that the tasks are achievable, but they require good depth perception and fast feedback. We use the secondary camera feed sent through the low-bandwidth channel to provide fast depth feedback. We found that the plug task requires a larger workspace than arm movement alone provides. At first, we let the robot sidestep while holding the plug, as shown in Figure 25. Later we utilize waist rotation to obviate the need to walk with the plug in hand.

During the Finals, the overhead lever pull and the plug moving were chosen as the surprise tasks for trial runs 1 and 2, respectively. The pull lever task was practiced in simulation and lightweight mockup only, but the plug moving task was practiced with a closely replicated setup, and we could consistently complete the task. Unfortunately, we did not have the chance to try the surprise task due to the fall for both the runs.

6.7. Rough Terrain and Debris

According to the DRC Finals rules, teams may choose either the Rough Terrain or Debris tasks to complete, and they can attain the same single point for either task. Although the details of the Rough Terrain task had been fairly well known before competition, it was not clear how the Debris task would be set up. Due to this uncertainty, we decided to try the Rough Terrain task.

To handle the rough terrain, we devised the extended locomotion controller using heel and toe lift. We set up the terrain in a simulated environment, shown in Figure 26, in order to test the ability of THOR-RD's range of motion and torque specifications. We found that the simulated robot can walk over the given terrain with a noisy surface model, and it does not have kinematic or joint torque issues. Without having extensively used the real THOR-RD robot on the terrain, we decided to skip both tasks and head to the well-tested Stairs task.

6.8. Stairs

We set up a mock staircase and tested walking up and down it with the robot, shown in Figure 27. To test the robustness of the climbing motion against the perception error, we set up each step height differently—the step heights are set to 23, 25, and 22 cm, respectively—as well as starting the robot in different initial positions. The robot could climb up and down the stairs with a high success rate in spite of incorrect surface models, thanks to the extended double support phase that uses heel and toe tilt with quasistatic stepping motions. Unfortunately, we did not progress to the Stair task due to the fall for both tasks.

6.9. Network Usage

Finally, to gauge robot autonomy and human interaction, we consider usage of the network channels. The upstream data capture the amount of information the human operator had to communicate to the robot. For the outdoor network conditions, we used 34 commands with 2,366 bytes to instruct the robot on how to open and go through the door. Inside, we used 59 commands with 3,344 bytes to approach and turn the valve and walk away. We omitted the driving task usage, as it was a very low-dimensional task, with an-

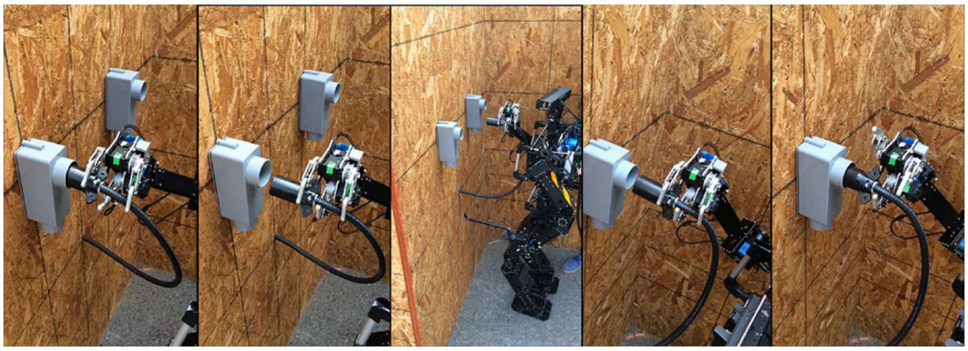


Figure 25. The robot pulls the plug out of the socket and mounts it in the other socket during the DRC Testbed.

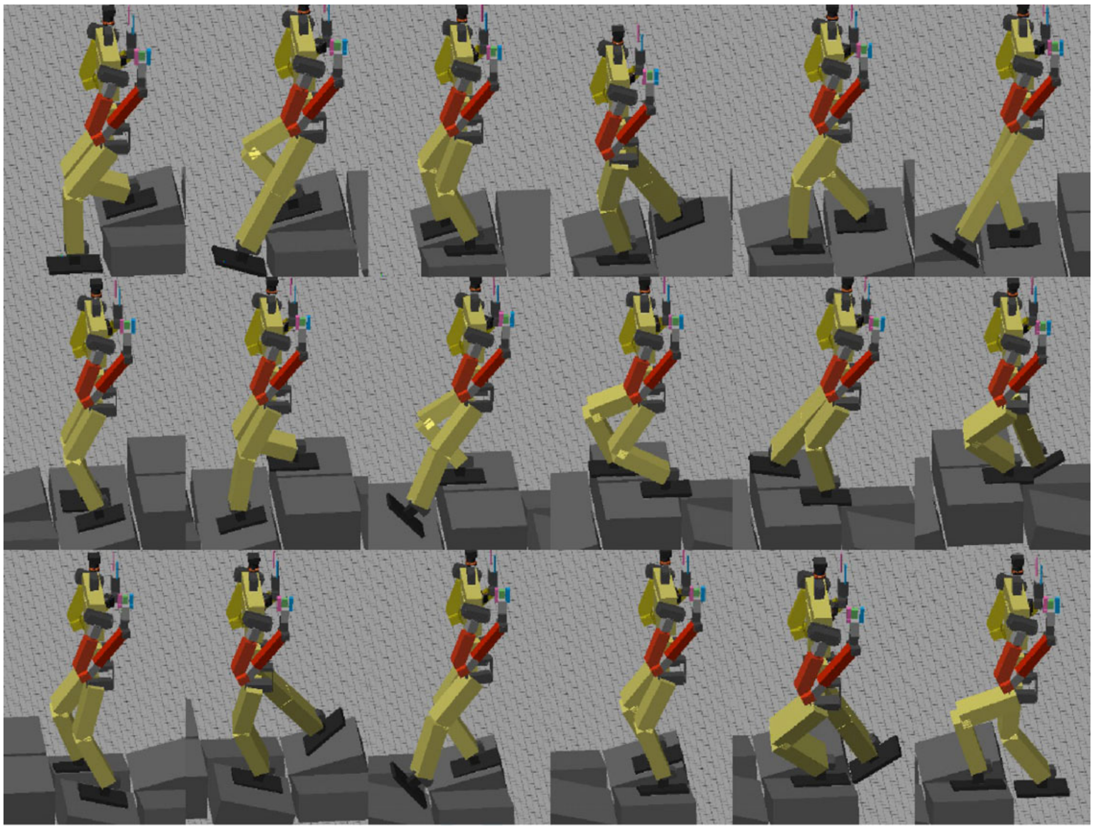


Figure 26. The THOR-RD robot traverses over the DRC Finals uneven terrain in a simulated environment.

gular changes in the wrist and head, and binary changes in the throttle; it was a very teleoperated activity with full availability of the high-bandwidth channel.

Shown in Figure 28 is the cumulative network usage over time for commanding the robot after the driving task. More important than the cumulative usage is the frequency of commands sent to the robot indoors. Long pauses in commands could indicate that too much time is taken by the operator to understand the scene. Figure 29 shows our intervals between successive commands.

The downstream data show how much information the human was able to process during an overall trial. Table II summarizes our network usage of downstream perception; the four perception items at the top were subject to network dropouts, while the bottom three utilized the low-bandwidth channel that had no dropouts. The number of bytes utilized will be different between runs because fewer packets are dropped later in the run. As we needed less time for our second run, we used less information to achieve the same number of points.

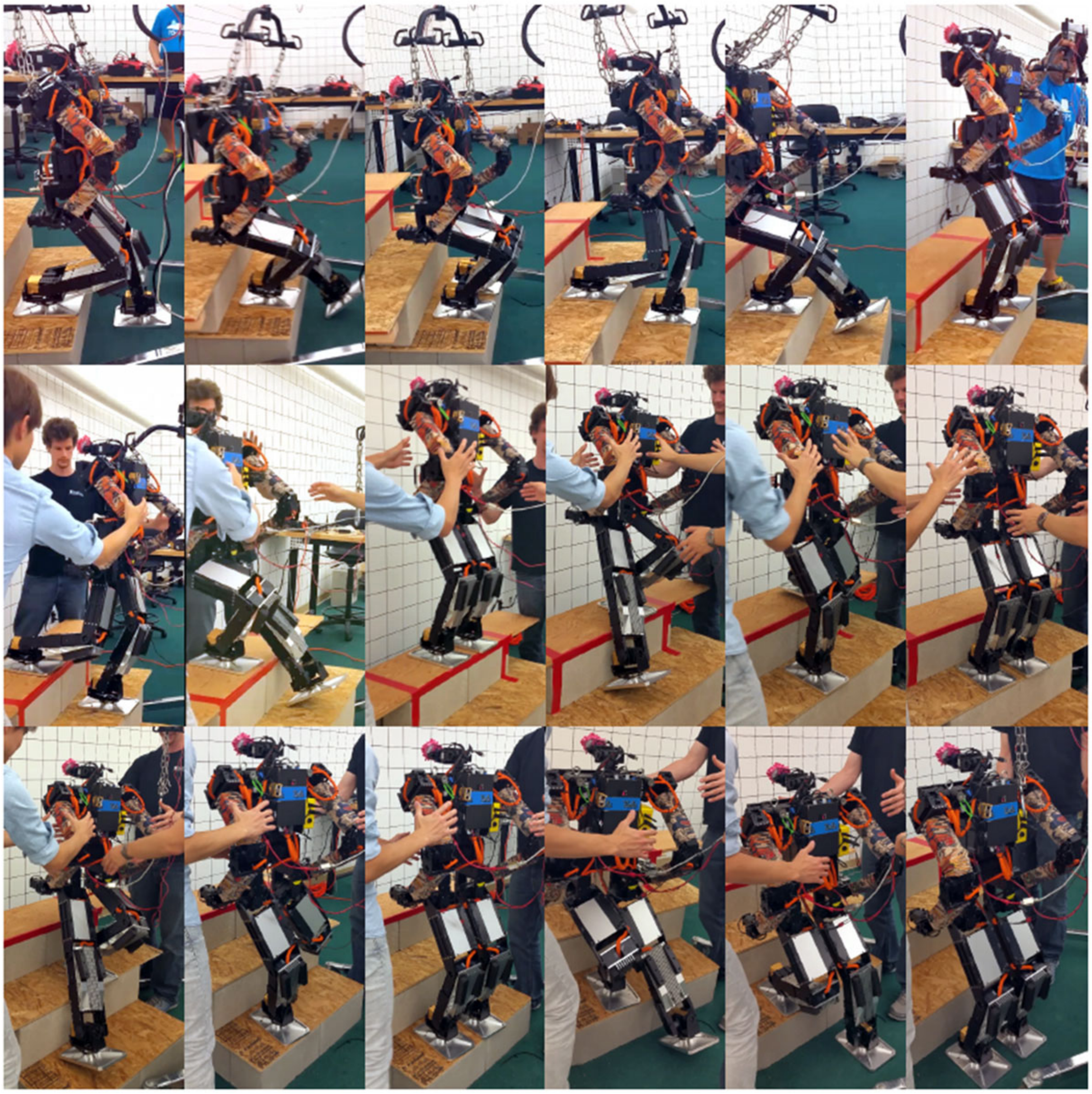


Figure 27. The THOR-RD robot climbing a set of stairs using the toe and heel lift controller.

7. LESSONS LEARNED

The DARPA Robotics Challenge allots a limited time frame for research teams to implement reliable systems that are robust to outdoor environments. To perform well, we managed our resources with priorities on rapid iterations and repeated system testing. In this section, we provide specific strategies that helped our overall effort, and we present our thoughts on improvements for research and development. Above all, we credit a supportive team that continually worked under pressure and across time zones to finish the DRC Finals.

7.1. Hardware Iterations

While major software changes can be implemented in short periods of time, hardware modifications require days to see even minor updates. We found that simultaneously fabricating design iterations on grippers and feet provided a time-efficient way to maximize our hardware setup. Simultaneous manufacturing meant trying several different fingers and proactively making new designs before testing on the real robot finished from previous designs. While newly fabricated hardware was not always tested thoroughly, this library of parts was invaluable. By the week

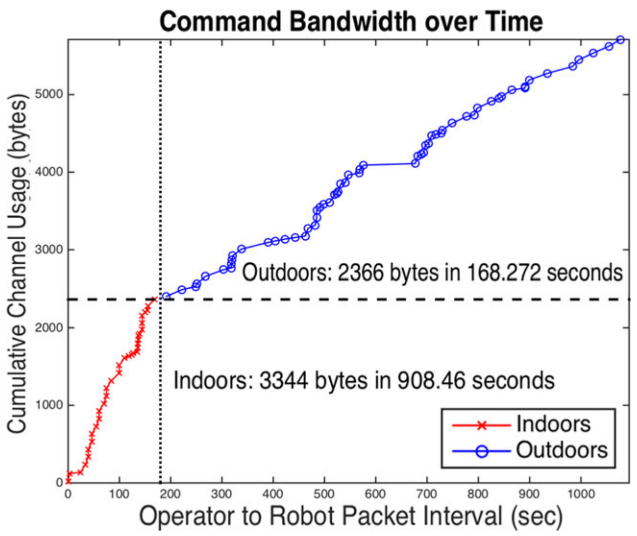


Figure 28. Operator command rates slowed down upon transitioning from the outdoor network conditions to indoor conditions.

Table II. Network usage (kilobytes) for Trials 1 and 2 indicate how much information was provided to the human.

	Outdoor	Indoor	Outdoor	Indoor
	High bandwidth			
ChestMesh	237,714	802,240	52,251	73,537
HeadMesh	45,062	396,985	3,082	32,921
HeadCam	152,104	102,995	34,464	8,172
HandCam	19,667	21,585	5,948	1,461
	Low bandwidth			
Feedback	363	294	77	214
HeadCam	0	1,071	0	629
HandCam	0	42	0	15

of the Finals, we had many different finger and arm combinations at our disposal in case we found one setup more suitable to the task configuration. Similarly, many foot choices were available, and we feel that this mix and match approach mimics real-world disaster response needs.

7.2. Network Testing

The Maxwell network emulator was able to drop packets to simulate real network blackouts; however, the implementation of the device differed somewhat from the rules description. UDP Packets larger than the MTU size of 1,500 were actually discarded, instead of packets larger than the 64k UDP packet size limit. Due to transmission fragmentation to 1,500 byte levels, a 64k packet was not allowed to pass by the network emulator. This important distinction meant that the network shakedown at the DRC Testbed in South Carolina became one of the most crucial aspects of our development. While we were able to code a packet reconstruction routine and send 1,500 byte fragments of 64k messages in South Carolina, purchasing the Maxwell Pro became a requirement. Testing with the emulator additionally allowed us to maximize the usage of the low-bandwidth channel in order to send image frames every 2 s. Over many tests with the robot, we calibrated the best JPEG quality settings, resolution, color space, and frame rate for both the head and wrist cameras. In a real disaster, the ability to calibrate these settings may be crucial.

7.3. Backup Robot

We focused significant energy on having two robots fully working at any given time—a luxury few teams enjoyed. We could test two different portions of software on a robot at the same time, which effectively doubled our development rate. Additionally, it eliminates downtime from hardware repair or reconfiguration.

Although we did not have two identical robots from the beginning, the fully modular nature of the robot allows for incremental upgrades—mixing and matching of different robot components was no hindrance to progress. We built two "Frankenstein" versions of the robot from one THOR-OP and one THOR-RD robot: one with powerful THOR-RD legs and the old THOR-OP upper body for locomotion testing, and one with the old THOR-OP legs and a new THOR-RD upper body for perception and manipulation testing. This mixture of parts required a very flexible configuration system in which kinematic changes, IMU device protocols, camera settings, etc. could be modified fluidly. We had another set of THOR-RD upper-body structural parts that we used to test add-on power and electronic components prior to putting them on the robot.

By the time of the Finals, we had two nearly identical robots fully assembled and tested. This proved very useful after our fall in the first trial. We swapped an entire leg very quickly, without needing to replace motors one at a time.

7.4. Calibration and Sensor-based Planning

With the availability of regular low-bandwidth sensory feedback, we did not prepare a rigorous calibration between the arm and leg joints of the robot and the mounted sensors. Paired with the multiple angle camera feeds, we found that a quick calibration of the LIDAR sensor is more than enough for most manipulation tasks, and we prefer to have extra confidence from visual feedback before executing motions.

Still, we spent considerable time on each manipulation task. A semiautonomous approach with well-calibrated sensory feedback could allow much faster operation in general, albeit with more risks. Similarly, we feel that Force-Torque sensor feedback added to our stepping strategy would provide much better modeling of the stairs and rough terrain, where touchdown information would inform LIDAR-only

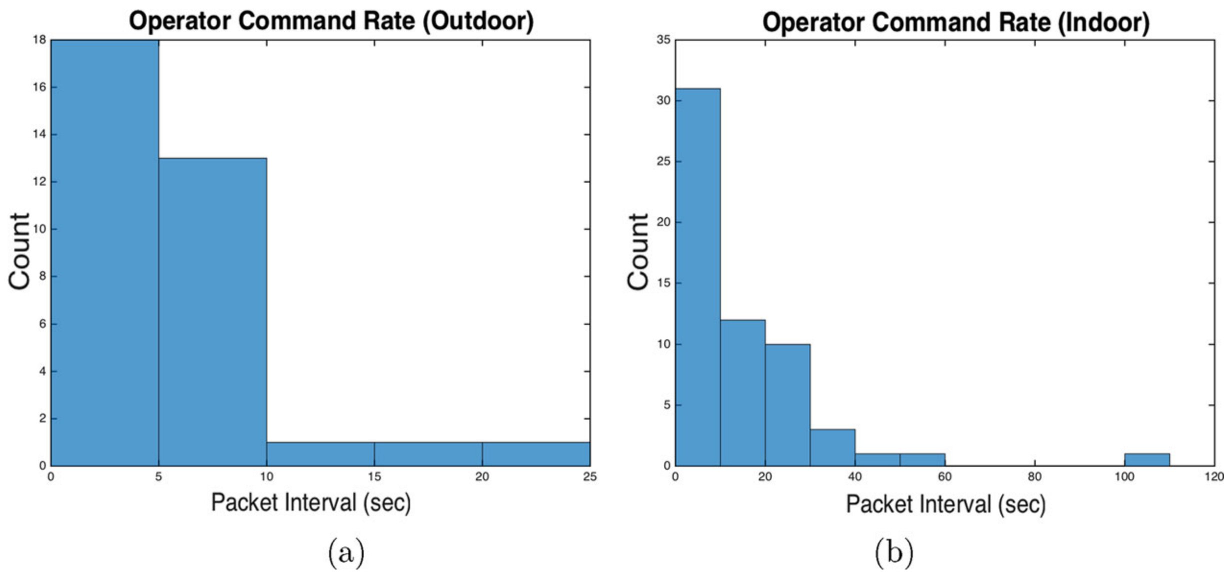


Figure 29. The indoor command rate spread out significantly as the latency between high-resolution information sent from the robot was increased from the outdoor network settings.

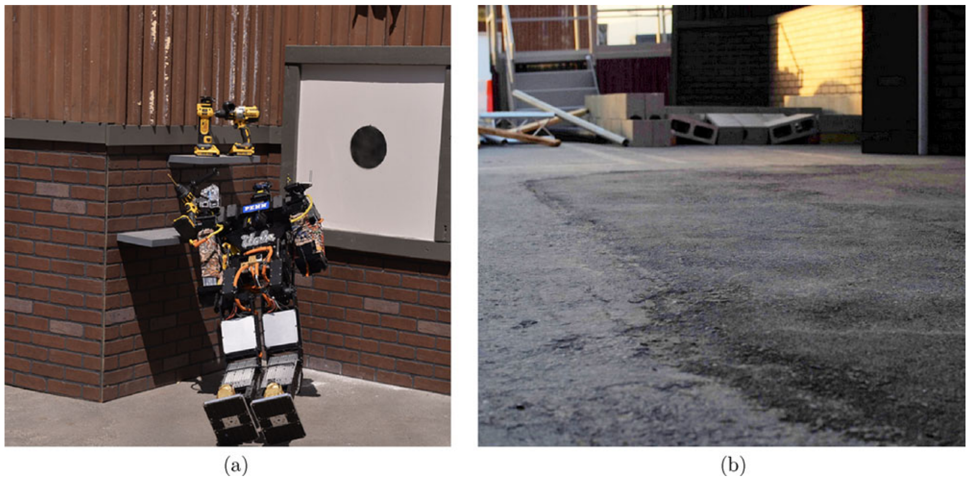


Figure 30. (a) The THOR-RD robot fell down after stepping on the surface irregularities. (b) The closeup of the surface shows irregularities due to patching of the asphalt.

models. In the future, calibration and sensor-based planning should become a high priority.

7.5. Environment Foreknowledge

The DRC testbed provided a very representative environment of the DRC Finals, and details about most tasks were very well known in advance. Teams were able to undergo extensive testing with replicas of the competition environment. This knowledge acts as a double-edged sword for the perceived outcomes of the DRC Finals. While the results of the winning teams were impressive, questions still remain about arbitrary environmental objects. However, with the observed spread in completion of manipulation tasks, it seems that giving *a priori* knowledge was prudent.

The most surprising part of the DRC Finals competition was the terrain. Just days before the event, the DRC officials revealed that the testing environment included a global downward slope of approximately 3.5% toward the task wall. The actual surface also had severe local irregularities, as shown in Figure 30. Negotiating this terrain, and adapting very quickly, is a true hallmark of disaster response, even if this unexpected environment did cause many robots, including ours, to fall.

We are excited about our accomplishments in motion planning, adaptive autonomy, and robust systems design. However, there is still some concern about how much design catered exclusively to the DRC and how relevant our methods will be in a real disaster where next to nothing is known about the environment.

8. CONCLUSIONS

The demanding DARPA Robotics Challenge Finals established a proving ground for the latest research into disaster response robotics. The competition pushed teams to focus on autonomous systems that cooperate with human operators, while overcoming unpredictable outdoor conditions that represent the real world. We presented Team THOR's particular approach to the challenge, which includes effective network usage strategies, planning routines that incorporate high-level human objectives, and modular hardware to traverse the terrain via bipedal locomotion.

In the DRC Trials, we showed that lightweight modular humanoid robots are ready to tackle the disaster response challenge. For the DRC Finals, we improved our hardware's reliability and functionality, supported extremely low bandwidth feedback for the operator interface, implemented a novel upper-body planner, and accommodated surface changes with a new locomotion strategy. Team THOR consistently scored three points in the final competition and reduced the task completion time between its two runs.

This consistency shows a level of robust behavior and highlights important areas for future research. We successfully split behaviors into upper-body and lower-body control, with little in the way of full-body motions. Intelligent full-body motions that respond dynamically to the environment remains a challenge. Having fallen in both of our runs, we are looking both at better surface adaptation strategies and hardware that can survive such impacts.

While we have identified such algorithmic and physical priorities, much of the DRC competition included human factors. Incorporating close to real-time feedback and allowing human operators to drive planning systems gave the operator a better sense of control and confidence in the robot. Future work for disaster response robotics in unknown environments will require this sense of confidence for timely execution.

ACKNOWLEDGMENTS

We acknowledge support from the Defense Advanced Research Projects Agency GS100000185 http://dx.doi.org/10.13039/100000185 (DARPA) through Grant No. GS100000185N65236-12-1-1002. We also acknowledge the support of the ONR SAFFIR program under Contract No. N00014-11-1-0074 and the ONR ARPIOS program under Contract No. N00014-15-1-2064.

REFERENCES

Burke, J. L., Murphy, R. R., Rogers, E., Lumelsky, V. J., & Scholtz, J. (2004). Final report for the DARPA/NSF interdisciplinary study on human-robot interaction. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 34(2), 103–112.

Buss, S. R., & Kim, J.-S. (2005). Selectively damped least squares for inverse kinematics. Journal of Graphics, GPU, and Game Tools, 10(3), 37–49.

Chang, P.-H. (1987). A closed-form solution for inverse kinematics of robot manipulators with redundancy. IEEE Journal of Robotics and Automation, 3(5), 393–403.

Chen, F., Taguchi, Y., & Kamat, V. R. (2014). Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In 2014 IEEE International Conference on Robotics and Automation (ICRA) (pp. 6218–6225).

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(8), 790–799.

Chiaverini, S. (1997). Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. IEEE Transactions on Robotics and Automation, 13(3), 398–410.

Cohen, B., Chitta, S., & Likhachev, M. (2013). Single- and dual-arm motion planning with heuristic search. The International Journal of Robotics Research, 33(2), 305–320.

Fankhauser, P., Bloesch, M., Rodriguez, D., Kaestner, R., Hutter, M., & Siegwart, R. (2015). Kinect v2 for mobile robot navigation: Evaluation and modeling. In 2015 International Conference on Advanced Robotics (ICAR) (pp. 388–394).

Goodrich, M. A., & Schultz, A. C. (2007). Human-robot interaction: A survey. Foundations and Trends in Human-computer Interaction, 1(3), 203–275.

Hebert, P., Bajracharya, M., Ma, J., Hudson, N., Aydemir, A., Reid, J., Bergh, C., Borders, J., Frost, M., Hagman, M., et al. (2015). Mobile manipulation and mobility as manipulation design and algorithms of robosimian. Journal of Field Robotics, 32(2), 255–274.

Heger, F. W., & Singh, S. (2006). Sliding autonomy for complex coordinated multi-robot tasks: Analysis & experiments. Robotics: Science and Systems.

Hollerbach, J. M. (1985). Optimum kinematic design for a seven degree of freedom manipulator. In Robotics Research: The Second International Symposium (pp. 215–222). Cambridge, MA: MIT Press.

Holz, D., Holzer, S., Rusu, R. B., & Behnke, S. (2011). Real-time plane segmentation using rgb-d cameras. In RoboCup 2011: Robot Soccer World Cup XV (pp. 306–317). Springer.

Jeong, H., & Lee, D. D. (2016). Learning complex stand-up motion for humanoid robots. In Proceedings of the 30th Association for the Advancement of Artificial Intelligence (AAAI 2016).

Johnson, M., Shrewsbury, B., Bertrand, S., Wu, T., Duran, D., Floyd, M., Abeles, P., Stephen, D., Mertins, N., Lesman, A., et al. (2015). Team IHMC's lessons learned from the

DARPA robotics challenge trials. Journal of Field Robotics, 32(2), 192–208.

Klein, C., & Huang, C.-H. (1983). Review of pseudoinverse control for use with kinematically redundant manipulators. IEEE Transactions on Systems, Man and Cybernetics, SMC-13(2), 245–250.

Kohlbrecher, S., Romay, A., Stumpf, A., Gupta, A., von Stryk, O., Bacim, F., Bowman, D. A., Goins, A., Balasubramanian, R., & Conner, D. C. (2015). Human-robot teaming for rescue missions: Team ViGIR's approach to the 2013 DARPA robotics challenge trials. Journal of Field Robotics, 32(3), 352–377.

Lloyd, J. E., & Hayward, V. (2001). Singularity-robust trajectory generation. The International Journal of Robotics Research, 20(1), 38–56.

Michel, O. (2004). Webots^{TM}: Professional mobile robot simulation. CoRR, abs/cs/0412052.

Murphy, R. R. (2015). Meta-analysis of autonomy at the DARPA robotics challenge trials. Journal of Field Robotics, 32(2), 189–191.

Na, M., Yang, B., & Jia, P. (2008). Improved damped least squares solution with joint limits, joint weights and comfortable criteria for controlling human-like figures. In 2008 IEEE Conference on Robotics, Automation and Mechatronics (pp. 1090–1095). IEEE.

Nakamura, Y., & Hanafusa, H. (1986). Inverse kinematic solutions with singularity robustness for robot manipulator control. Journal of Dynamic Systems, Measurement, and Control, 108(3), 163–171.

Nishiwaki, K., Chestnutt, J., & Kagami, S. (2012). Autonomous navigation of a humanoid robot over unknown rough terrain using a laser range sensor. The International Journal of Robotics Research, 31(11), 1251–1262.

Rouleau, M., & Hong, D. (2014). Design of an underactuated robotic end-effector with a focus on power tool manipulation. In ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (pp. V05BT08A027–V05BT08A027). American Society of Mechanical Engineers.

Schulman, J., Ho, J., Lee, A., Awwal, I., Bradlow, H., & Abbeel, P. (2013). Finding locally optimal, collision-free trajectories with sequential convex optimization. In Proceedings of Robotics: Science and Systems. Berlin.

Shimizu, M., Kakuya, H., Yoon, W.-K., Kitagaki, K., & Kosuge, K. (2008). Analytical inverse kinematic computation for 7-dof redundant manipulators with joint limits and its application to redundancy resolution. IEEE Transactions on Robotics, 24(5), 1131–1142.

Siciliano, B. (1990). Kinematic control of redundant robot manipulators: A tutorial. Journal of Intelligent and Robotic Systems, 3(3), 201–212.

Slotine, J.-J. E. (1985). The robust control of robot manipulators. The International Journal of Robotics Research, 4(2), 49–64.

Stentz, A., Herman, H., Kelly, A., Meyhofer, E., Haynes, G. C., Stager, D., Zajac, B., Bagnell, J. A., Brindza, J., Dellin, C., et al. (2015). CHIMP, the CMU highly intelligent mobile platform. Journal of Field Robotics, 32(2), 209–228.

Weghe, M. V., Ferguson, D., & Srinivasa, S. S. (2007). Randomized path planning for redundant manipulators without inverse kinematics. In 2007 7th IEEE-RAS International Conference on Humanoid Robots (pp. 477–482). IEEE.

Yanco, H. A., Norton, A., Ober, W., Shane, D., Skinner, A., & Vice, J. (2015). Analysis of human-robot interaction at the DARPA robotics challenge trials. Journal of Field Robotics, 32(3), 420–444.

Yi, S.-J., Hong, D., & Lee, D. D. (2013). A hybrid walk controller for resource-constrained humanoid robots. In 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids).

Yi, S.-J., McGill, S., He, Q., Vadakedathu, L., Yi, H., Cho, S., Hong, D., & Lee, D. D. (2015). Robocup 2014 humanoid adult size league winner. In RoboCup 2014: Robot World Cup XVIII (pp. 94–105). Springer.

Yi, S.-J., McGill, S. G., Vadakedathu, L., He, Q., Ha, I., Han, J., Song, H., Rouleau, M., Zhang, B.-T., Hong, D., et al. (2014). Team THOR's entry in the DARPA robotics challenge trials 2013. Journal of Field Robotics, 32(3), 315–335.

Zucker, M., Joo, S., Grey, M. X., Rasmussen, C., Huang, E., Stilman, M., & Bobick, A. (2015). A general-purpose system for teleoperation of the DRC-HUBO humanoid robot. Journal of Field Robotics, 32(3), 336–351.