# Team THOR's Entry in the DARPA Robotics Challenge Trials 2013

**Seung-Joon Yi**
*GRASP Lab, University of Pennsylvania, Philadelphia, Pennsylvania 19104*
*e-mail: yiseung@seas.upenn.edu*
**Stephen G. McGill**
*GRASP Lab, University of Pennsylvania, Philadelphia, Pennsylvania 19104*
*e-mail: smcgill3@seas.upenn.edu*
**Larry Vadakedathu**
*GRASP Lab, University of Pennsylvania, Philadelphia, Pennsylvania 19104*
*e-mail: vlarry@seas.upenn.edu*
**Qin He**
*GRASP Lab, University of Pennsylvania, Philadelphia, Pennsylvania 19104*
*e-mail: heqin@seas.upenn.edu*
**Inyong Ha**
*Robotis, Seoul, Korea*
*e-mail: dudung@robotis.com*
**Jeakweon Han**
*Robotis, Seoul, Korea*
*e-mail: jkhan@robotis.com*
**Hyunjong Song**
*Robotis, Seoul, Korea*
*e-mail: hjsong@robotis.com*
**Michael Rouleau**
*RoMeLa, Virginia Tech, Blacksburg, Virginia 24061*
*e-mail: mrouleau@vt.edu*
**Byoung-Tak Zhang**
*BI Lab, Seoul National University, Seoul, Korea*
*e-mail: btzhang@bi.snu.ac.kr*
**Dennis Hong**
*RoMeLa, University of California, Los Angeles, Los Angeles, California 90095*
*e-mail: dennishong@ucla.edu*
**Mark Yim**
*GRASP Lab, University of Pennsylvania, Philadelphia, Pennsylvania 19104*
*e-mail: yim@seas.upenn.edu*
**Daniel D. Lee**
*GRASP Lab, University of Pennsylvania, Philadelphia, Pennsylvania 19104*
*e-mail: ddlee@seas.upenn.edu*

Received 9 March 2014; accepted 29 September 2014

This paper describes the technical approach, hardware design, and software algorithms that have been used by Team THOR in the DARPA Robotics Challenge (DRC) Trials 2013 competition. To overcome big hurdles such as a short development time and limited budget, we focused on forming modular components—in both hardware and software—to allow for efficient and cost-effective parallel development. The hardware of THOR-OP (Tactical Hazardous Operations Robot–Open Platform) consists of standardized, advanced actuators and structural components. These aspects allowed for efficient maintenance, quick reconfiguration, and most importantly, a relatively low build cost. We also pursued modularity in the software, which consisted of a hybrid locomotion engine, a hierarchical arm controller, and a platform-independent remote operator interface. These modules

 
View this article online at wileyonlinelibrary.com • DOI: 10.1002/rob.21555

yielded multiple control options with different levels of autonomy to suit various situations. The flexible software architecture allowed rapid development, quick migration to hardware changes, and multiple parallel control options. These systems were validated at the DRC Trials, where THOR-OP performed well against other robots and successfully acquired finalist status. 

## 1. INTRODUCTION

One of the goals of the DARPA Robotic Challenge[1] (DRC) is to bring together robotics experts to make a significant breakthrough in the field of humanoid robotics. There were few attempts to build a robust, complete, remotely operated humanoid robot system for practical tasks due to a number of reasons: the high cost of developing and maintaining a high degree of freedom (DOF) robot, the difficulty of developing a robust locomotion engine, the risk associated with operating a statically unstable robot in uncontrolled environments, and the lack of clear motivation. The DRC provides an opportunity to overcome this situation and is furnishing a capable common humanoid platform[2] and providing research funding to selected teams, as well as worthy prizes for the winning teams. Figure 1 shows THOR-OP (Tactical Hazardous Operations Robot Open Platform), Team THOR's attempt at designing, building, and programming a robust low cost, modular disaster response humanoid robot. In this paper, we provide an overview of the DRC Trials and describe the technical approach of Team THOR in detail.

### 1.1. Problem Statement

The DRC specifically requires a complete system that has the mobility, dexterity, strength, and platform endurance required for a practical disaster response situation. More specifically, the robot should be able to operate in unstructured environments that include piles of blocks, ladders and doorways, industrial valves, power tools, and vehicles. Tasks in this operation domain require full body control and cohesion among software components hitherto not seen before. In addition, only a limited bandwidth is allowed for communication between the robot and the operator, which penalizes teams who rely heavily on external computation or direct teleoperation of the robot. Such requirements pose a great challenge for hardware and software; an especially short preparation time for the DRC Trials further mandates a limited development period allowed for each team.

### 1.2. Scoring and Ranking

The 2013 DRC Trials consisted of eight separate tasks that required the robot to perform a variety of demanding tasks, from maneuvering through doors to operating a power drill to driving a vehicle. As all tasks were performed in unknown outdoor conditions, the robot was required to be skillful and robust.

Each task was broken into three subtasks; one point is awarded for the completion of each subtask. If the robot completes all three tasks without human intervention, teams are awarded another bonus point. Teams are ranked primarily according to the number of points accrued over all eight tasks. Tie-breaks are done first using the total number of interventions (times in which a human could physically interact with the robot), and then the total time spent.

### 1.3. Challenges

The competition is challenging as teams must overcome a number of unsolved problems in humanoid robotics fields with a short preparation time. Among many technical issues, we view the following problems as the major challenges for the competition:

- Building a reliable and capable hardware platform
- Robust locomotion in an unstructured environment
- Dexterous bimanual manipulation
- Sliding autonomy for controlling a robot
- Bandwidth-limited remote user interface

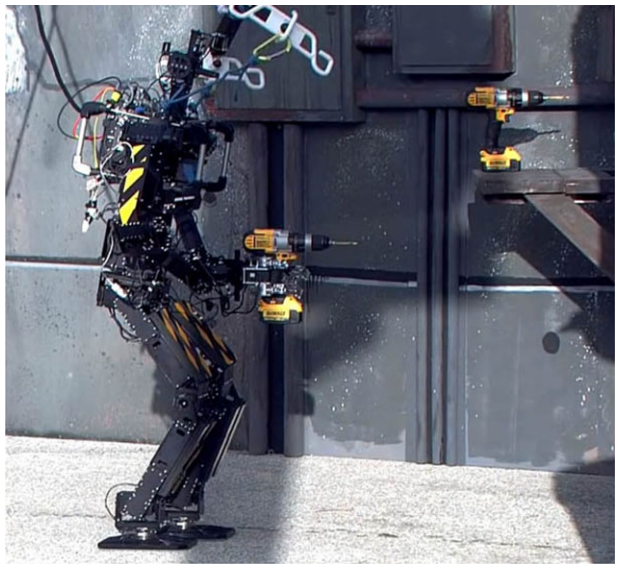


**Figure 1.** The THOR-OP robot approaching the wall to break while holding a drill in its hand.

[1]DARPA Robotics Challenge. http://go.usa.gov/mVj
[2]Atlas—The agile anthropomorphic robot, http://www.bostondynamics.com/robotAtlas.html

We describe in more detail how we have addressed each technical difficulty in the following sections. With the short time frame and varied operation requirements, we focus development on modularity and cost effectiveness.

#### 1.3.1. Hardware

Our hardware is adaptive in that all of the actuators and structural parts are modular elements developed for the commercial market. The robot is assembled simply by bolting those advanced series of actuators and custom mounting brackets together, allowing us to keep development costs low and reduce manufacturing and repair times. The whole robot can be assembled from machined parts in only 24 man hours, and a damaged actuator can be swapped out on the field within several minutes. With commercial off-the-shelf parts,[3] THOR-OP is one of the least expensive humanoid robots at the DRC Trials.

#### 1.3.2. Software

We have been developing our modular humanoid robot software framework over the course of several years (McGill et al., 2010), using it successfully in the humanoid robotic soccer domain. It has been tested with different humanoid robots including the Nao (Gouaillier et al., 2008), the DARwIn series (Muecke & Hong, 2007), and CHARLI (Han, 2012). Due to the extensible nature of the framework, we are able to quickly port the code to new humanoid robots by writing only a new interface module and kinematic description. Similarly, we simulate new robots using the Webots robotics simulator (Michel, 2004) with minimal effort.

In general, we divide the software development into smaller, reusable projects that can be tested individually to allow for parallel development without complete knowledge of the whole system. This reusable and easily extensible nature also drives our interprocess software stack, where multiple methods access the sensor feeds and robot commands. Finally, we have developed multiple hierarchical modules for locomotion, manipulation control, and the operator interface that can be dynamically selected to suit the situation at hand.

## 2. HARDWARE ARCHITECTURE

For most high-load joints, we use the new PRO series of Dynamixel actuators developed by Robotis, Co. Ltd. We use three different Dynamixel Pro actuator types: H42-20-S300-R, H54-100-S500-R, and H54-200-S500-R. They are rated at 20, 100, and 200 W, respectively, and can be fitted with a number of different reduction gear boxes.

For the THOR-OP platform, two different gearboxes are used, one with an in-line output axle and one with a parallel output axle. Both gearboxes use cycloidal reduction gears that have a higher impact tolerance than common harmonic drives. They utilize 4,096-step absolute encoders (after gear reduction), which enable precise control. With a maximum of 502,000 counts per revolution, the user can measure joint angles to 0.0007 degrees, allowing for high precision control. The actuators communicate over a serial bus, where a number of actuators can be connected in a daisy chain setup. The actuators can be commanded through position, torque, and speed with electrical current sensing based control. They are certified by the Korea Testing Laboratory and the Korea Measuring Instrumentation Research Association to the specifications provided by the manufacturer.[4]

**Table I.** Specifications comparison of tracks A and B robots at DRC Trials 2013.

| Robot | Weight (kg) | Height (cm) | Wingspan (cm) |
|---|---|---|---|
| CHIMP | 181 | 157 | 305 |
| Atlas | 150 | 188 | 260 |
| Valkyrie | 130 | 188 | 203 |
| RoboSimian | 108 | 164 | 221 |
| Schaft | 95 | 148 | 262 |
| HUBO | 60 | 140 | 204 |
| THOR-OP | 49 | 147 | 195 |

### 2.1. Mechanical Design

The THOR-OP robot consists of 31 Dynamixel actuators, 7 in each arm, 6 in each leg, 2 in the torso, 2 for the head, and 1 for panning the chest LIDAR (Light Detection and Ranging). The robot stands 1.47 m tall, weighs 49 kg, and has a wingspan of approximately 1.95 m.

Table I shows the specification of all track A and track B robots that participated at the DRC Trials, where our robot was one of the lightest and smallest robots. Despite the lightweight and compact design, the robot was capable of performing all of the heavy duty tasks for the DRC Trials, which includes manipulating heavy power tools, driving a vehicle, and walking over uneven terrain reasonably well. Figures 2 and 3 show the detailed shape, linkage dimensions, and kinematic configuration of the robot.

#### 2.1.1. Lower Body Design

The lower body consists of a pelvis and legs, which have a total of 12 Dynamixel Pro actuators. We use the conventional 6 DOF leg design used in most of the current humanoid robots (Ishida, 2004; Kaneko et al., 2004; Kim et al., 2012; Park, Kim, Lee, and Oh, 2007; Sakagami et al., 2002); it has

[3]http://www.robotis-shop-en.com/

[4]Manufacturer Specifications of Dynamiel Pros, http://www.robotis.com/xe/DynamixelPro_en

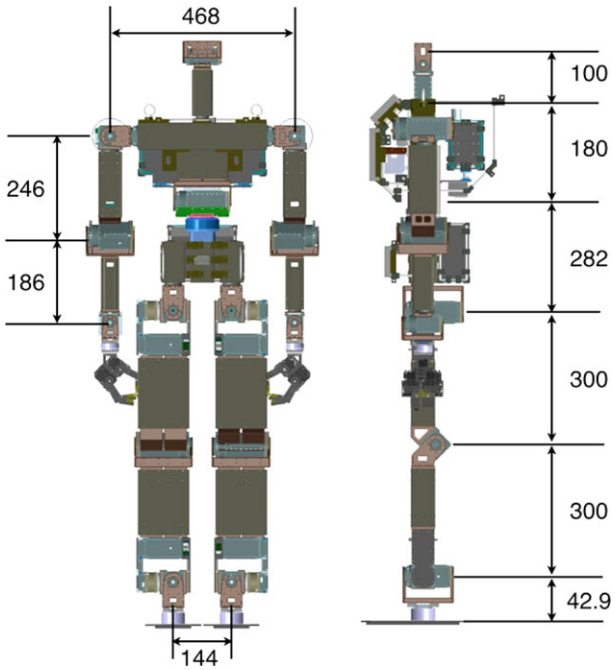



**Figure 2.** Front and side views of the THOR-OP robot with dimensions (units of mm). The arms shown have six DOFs.

three colocated hip joints, one knee joint, and two colocated ankle joints. Our legs are distinctively wider and thinner than those of other robots due to the shape of the modular actuator we use for the legs, and the knee joints have an offset to achieve a full 180 degrees range of motion (ROM). Thanks to the leg shape and the extensive knee ROM, the dismantled leg can be fully folded and stored in a small suitcase for easy transport. The foot has gone through many iterations during development, and the current design has fixed ridges for strength and stiffness. The front of the legs is covered with thick urethane padding for protection from a possible fall or knee strike during locomotion.

#### 2.1.2. Upper Body Design

The upper body consists of two arms for manipulation, a sensor head and a panning chest LIDAR for perception, and onboard computers for computation. Our initial arm design was a conventional 6 DOF one (Park et al., 2007), which has three colocated shoulder joints for pitch, roll, and yaw, one joint for an elbow, and two colocated wrist joints. During testing, we found that the 6 DOF arm has a smaller workspace than required, which gets even worse with torso movement based balancing control. Thus, we have revised the arm with three colocated wrist joints. Instead of a more common yaw-roll-pitch wrist joint (Lim et al., 2012; Ogura et al., 2006), we chose a yaw-roll-yaw configuration, as shown in Figure 5, which has a cleaner

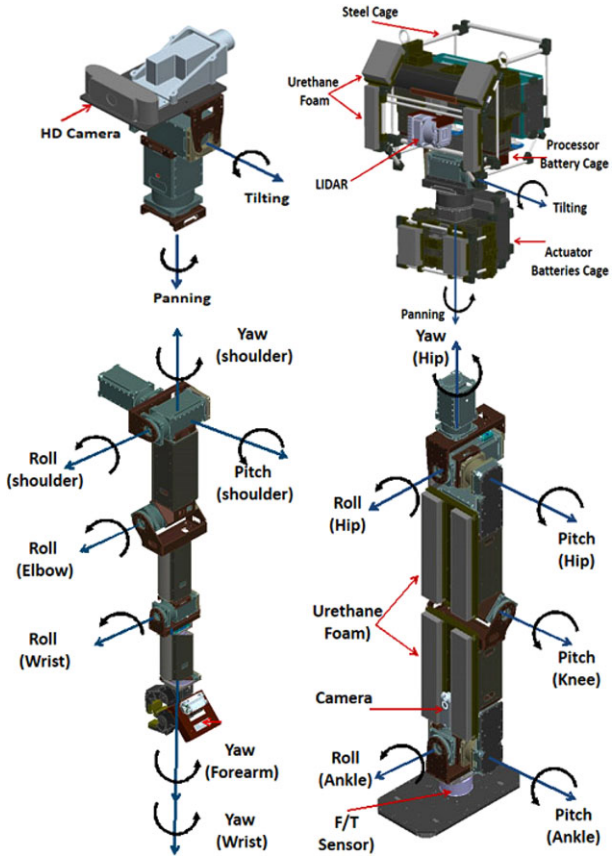



**Figure 3.** Kinematic configuration of THOR-OP.

structure and has a big advantage when the robot is rotating an object, such as turning a doorknob. Finally, we extended the arm length for an even larger workspace, as shown in Figure 5.

The torso section houses the panning chest LIDAR, onboard computers, and battery compartments; it is connected to the pelvis with two waist joints. The battery compartment is located close to the overall center of mass (COM) so that the robot can work stably with or without batteries. As it houses the main sensory, computation, and power components, it has a steel roll cage that doubles as gripping handles for additional protection. Above the torso, we have a simple sensor head connected by 2 neck joints. Table II enumerates the name, power rating, and range of motion for all the joints of the THOR-OP robot.

### 2.2. Modular Structural Components

In addition to the actuators, the robot is mainly built with standardized structural components that are designed to be used with the Dynamixel Pro actuators. They are extruded aluminum tubings and brackets with regularly spaced bolt holes, and one can easily assemble them with hex bolts.

**Table II.** Specifications of all THOR-OP joints.

| | Joints | Power Rating (Watts) | Range of Motion (Degrees) |
|---|---|---|---|
| **Head** | Head Pitch | 20 | −90 to 90 |
| | Head Yaw | 20 | −180 to 180 |
| **Torso** | LIDAR Pan | 16 | −45 to 45 |
| | Waist Pitch | 200 | −15 to 90 |
| | Waist Yaw | 200 | −90 to 90 |
| **Arm** | Shoulder Pitch | 200 | −180 to 180 |
| | Shoulder Roll | 200 | 0 to 180 |
| | Shoulder Yaw | 100 | −180 to 180 |
| | Elbow Pitch | 100 | 0 to 160 |
| | Forearm Yaw | 20 | −180 to 180 |
| | Wrist Yaw | 20 | −180 to 180 |
| | Wrist Roll | 20 | −90 to 90 |
| **Leg** | Hip Yaw | 100 | −60 to 55 |
| | Hip Roll | 200 | −90 to 35 |
| | Hip Pitch | 200 | −90 to 35 |
| | Knee Pitch | 200 | 0 to 180 |
| | Ankle Pitch | 200 | −90 to 35 |
| | Ankle Roll | 100 | −90 to 35 |

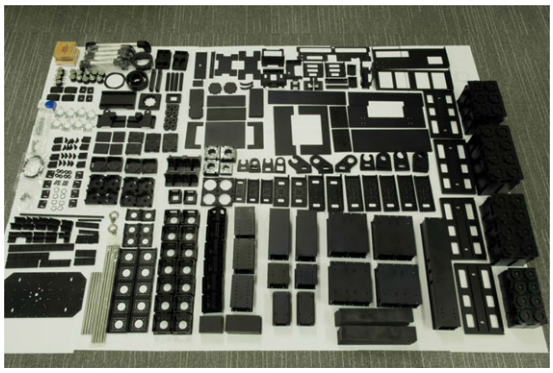


**Figure 4.** The structural components for a single THOR-OP robot use standardized dimensions.

Figure 4 shows the structural components required for a single THOR-OP robot; the total man hours needed for a complete assembly from parts is estimated at 24 h. With the benefit of the modular construction of the robot, we could quickly iterate through a number of different designs. Figure 5 shows the evolution of arm design over the course of our testing. The evolution of the arm DOF and link lengths was completed in a short time due to the easily adaptable design.

## 2.3. End Effector

One of the few nonstandard parts we use for the robot is the end effector. Over the course of our preparation for the DRC Trials, we designed and iterated multiple gripper designs.

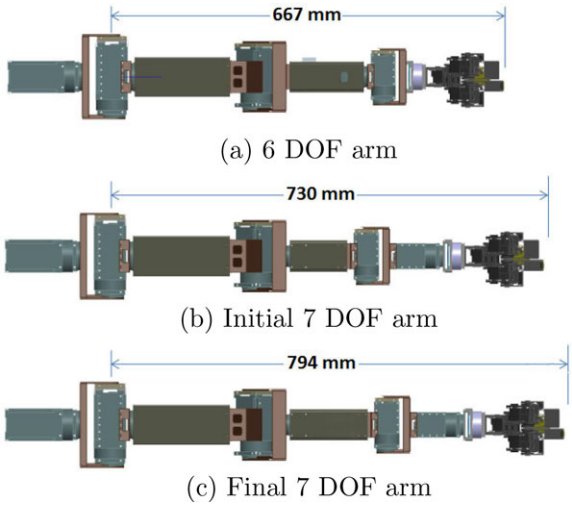



**Figure 5.** Three different arm configurations evolved during development of the THOR-OP.

The final design incorporates two active fingers controlled by smaller Dynamixel MX-106R actuators and a passive palm at the opposing side, shown in Figure 6(a).

### 2.3.1. Underactuated Finger Mechanism

Each finger has a passive joint with a spring-loaded linkage mechanism (Laliberte, Birglen, and Gosselin, 2002), as shown in Figure 6(b). When the finger comes into contact with an object while closing, the mechanism activates the second joint, making the finger wrap around the object (Rouleau & Hong, 2014). In practice, our hand can securely grip a wide range of objects, including drills, hoses, and wooden blocks, while being lightweight at only 797 g.

### 2.3.2. Passive End Effectors

The rules of the DRC Trials specify that the robot may use multiple end-effectors to suit different tasks as long as the robot carries all of them throughout the challenge. We used a number of passive end-effectors for tasks that do not require finger actuation, mainly to protect the finger mechanism from possible damage. We use three different types of appendages: straight rods for rotating valves and steering wheels, hooks for opening and pulling the doorknob, and a gear-shaped star for tightening the hose tip. They are designed to be robust against possible misalignment, which allows us to save time needed for fine positioning. Also, we have designed the appendages to be mounted on the side of the wrist, so that we do not have to detach the whole hand and let the robot carry them around.

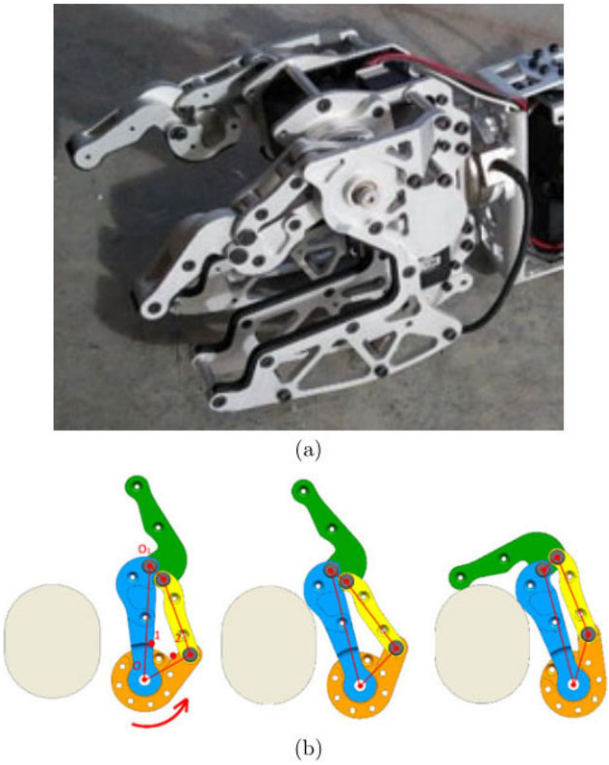



**Figure 6.** (a) The gripper for the DRC Trials consisted of two underactuated fingers to wrap around an object, and a rigid palm to give support. (b) The underactuated mechanism in the finger.

## 2.4. Electronic and Power Systems

For the onboard computation, the THOR-OP robot has two Axoimtek computers with a 1.65 GHz dual core AMD G-series APU. Communication from these computers to the actuators goes through a USB interface board and then is divided into four independent RS485 chains. As each of our modular actuators has built-in motor controllers and can be connected in a daisy chain network, our control setup was simple. Due to the simplicity and efficiency of our software, our computation load is light and we choose not to use a field computer for the competition, instead relying on a single onboard computer. Although this computer is not as powerful as some external computation, it proved capable enough for our approaches. During testing, computation was concentrated in the state machine process, which handled both the walk engine and arm planning. Total CPU utilization was less than 40%. Memory usage was quite low, and the six processes together occupied less than 10% of onboard memory.

The robot operates on 24 V for the onboard computer and actuators, except for the LIDAR and its panning actuator, which utilizes downconverted 12 V. The computer

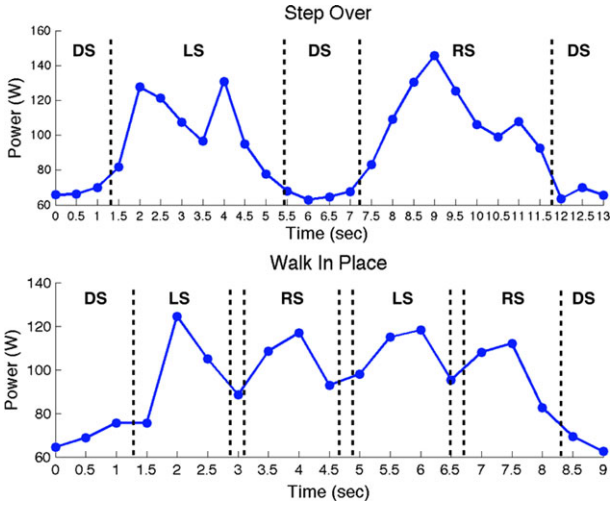



**Figure 7.** The power consumption varies during different phases of a locomotion cycle. The top shows when THOR-OP is taking a large step over a threshold 2.5 in. high, while the bottom represents walking in place (**DS**: double support, **LS**: left support, **RS**: right support).

power and the actuator power reside on a separate supply to mitigate harmful current spike effects. THOR-OP has three LiPo battery compartments, which can power the computer, upper body, and lower body in parallel to make the robot operate completely on battery power. However, as the DRC Trials did not require completely untethered operation, we used an external power supply for actuators, and we used the battery to just power the onboard computers. Overall, the robot is fairly efficient, consuming less than 100 W of power for most cases, with peak power consumption never exceeding 480 W. Figure 7 shows the power consumption of the robot for two different locomotion scenarios.

## 2.5. Sensory Components

The THOR-OP robot has a broad range of sensors. On the head, one Logitech C920 HD ProWebcam USB camera provides up to HD video coupled with a stereo microphone. To provide visual information from different perspectives, each wrist is equipped with a Logitech C905 Webcam. During testing, we evaluated an ultra-wide-angle USB camera to provide additional situational awareness, but we chose not to use it, as the specific unit occupied the full USB bus and seriously affected the whole system performance.

The robot is equipped with two ethernet-based Hokuyo UTM-30LX-EW LIDAR sensors, one on the head and one in the chest. We tested outdoors and found that the LIDARs are not affected by direct sunlight even without any special shades or covers. The chest LIDAR mounted vertically on a yawing actuator is used to generate a local three-dimensional (3D) representation of the surroundings

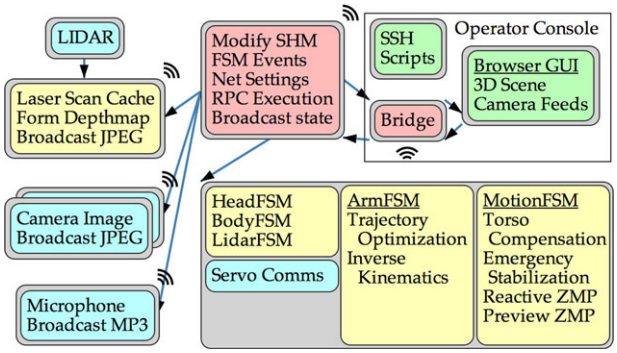



**Figure 8.** The overall control system layout provides important separation of modules. Blue represents a device interface, yellow represents autonomous logic, red indicates a signaling pathway, green shows a user interface, and gray borders represent separate processes.

for manipulation. From our testing experience, we were confident that the 3D mesh from the chest LIDAR provided accurate enough information for the robot to localize objects and navigate to target positions.

The head LIDAR, mounted horizontally, is mainly used to generate a 2D map using the simultaneous localization and mapping (SLAM) algorithm described in Butzke et al. (2012), however this is not utilized since all tasks are performed in a small-scale environment setup, where mapping is not necessary. Also, using one sensor instead of two reduced the bandwidth usage, which was advantageous given the degraded network condition in the DRC trials.

We used a Microstrain 3DM-GX3-45 inertial sensor located close to the center of mass of the robot. For pose estimation, we utilized its raw accelerometer and gyro data and its extended Kalman filtered inertial estimates.

## 3. SOFTWARE ARCHITECTURE

There are many advantages to modular software design. Each module performs logically discrete functionality, built separately from each other. When assembled together in the proper hierarchy, they constitute the application program. This type of system is reusable and extensible; in fact, we reused many modules from our own open-source code base for RoboCup (McGill et al., 2010) to save time and effort. Overall, we utilized C/C++ for low-level drivers, Lua for high-level state machines and libraries, and JavaScript and HTML5 for the user interface. Where heavy mathematical calculations were needed in Lua, we utilized the Torch7 library (Collobert, Kavukcuoglu, and Farabet, 2011) and its C API.

### 3.1. Core API

To achieve the modular design shown in Figure 8, each module needs to be able to communicate with the others. We use a combination of Boost managed shared memory segments[5] and ZeroMQ[6] channels. A remote procedure call system allows us to access shared memory through a NodeJS[7] server.

In addition to the communication core, we also have an abstract API that can work with a broad range of simulated or physical humanoid robot platforms. To do this, we maintain a number of configurations that store robot-specific parameters, such as sensor parameters, joint offsets, and walking parameters. Generic robot commands and queries are thus translated into robot-specific kinematics and motor packets, with the intention that a new robot platform needs only a similar module conforming to the API. In fact, we have the configuration for the simulated ATLAS model, which has helped us to revise our own hardware configurations during the early stage of development.

### 3.2. Sensor System

We set aside separate processes to interface each sensor: cameras, laser scanners, inertial measurement unit (IMU), and microphone. The camera and microphone information is directly sent over the noisy network to the human operator, compressed over a user-selectable user datagram protocol (UDP) or transmission control protocol (TCP) connection. The laser scanner information and appropriate metadata are sent over a reliable channel to the processes in an onboard computer, which includes the mesh-building process, which accumulates them to build a complete 3D scan of surroundings, and the SLAM process, which runs a 2D SLAM using the raw LIDAR data. The IMU readings, on the other hand, are set directly in shared memory.

### 3.3. Motion Control Architecture

Our motion control ranges from low-level topology of motor connections to high-level joint limits and human-guided motor information requests. With a well-connected hierarchy of control, a human can leverage both high-level commands and access low-level data, depending on the need.

#### 3.3.1. Low-level Motor Access

The THOR-OP robot has a total of 31 Dynamixel actuators that can communicate over a serial bus, connected in a daisy chain setup. In the base case, we can set a motor command or read current status for a single actuator, which requires a long time to complete if we iterate on all actuators in sequence. For this reason, the actuators support a synchronized command structure where a single packet can read or write multiple actuators at once. To further speed

[5]Boost c++ libraries. http://www.boost.org
[6]P. Hintjens (2010). ZeroMQ: The Guide. http://zguide.zeromq.org/page:all
[7]Node.js webserver. http://www.nodejs.org

up the communication, we use four daisy chains to handle four limbs in parallel. Extra actuators from the head, chest, and waist are divided appropriately onto each chain. Additionally, separating the chains yields better electrical characteristics and fewer timeouts due to packet loss based on our experiments.

We use a conservative approach of not reading from the servos at all except for the initialization process, to make sure that the communication speed is not slowed down by corrupt readings either from a damaged actuator or a bad connection. We would like to increase our performance and ability in reading motor states.

#### 3.3.2. Motor Manager

On the computer, we establish a dedicated process to communicate with actuator buses of the robot, which we called the motor manager. The role of this motor manager is to read commands from a shared memory segment, and to relay these commands to the motors. If the shared memory indicates a read operation request, then it will enqueue this instruction as well and receive the data packet from motors and write back to a shared memory segment. The motor manager also clamps commanded angles between an admissible range to prevent joint-level self-collisions.

#### 3.3.3. Shared Memory Access

We use shared memory throughout our software system, including our motor manager. This means that separate processes can control the motor commands and request motor information without any special broker and without resorting to running a fully integrated motion process. Practically, we are able to use a separate upper body motion controller and lower body motion controller to control the arms, neck, and waist and to control the leg joints, respectively. The two controllers function separately, which enables the robot to locomote during manipulation, yet they are coupled so that the robot can stay balanced in spite of different arm configurations.

### 3.4. Communication Architecture

In the DRC Trials, the bandwidth and latency over the IP network alternate each minute between good and bad conditions. The good communication condition is 1 Mbps bandwidth paired with 100 ms round trip delay, while bad communication is 10 Kbps paired with 1,000 ms. Due to the expectation of changing and limited available bandwidth, we allow for operator-specified on-the-fly configurable compression techniques and transmission frame rates of camera and LIDAR data.

In addition to raw sensor data encoding, we need to transmit metadata and other structured information. As our framework relies heavily on two different scripting languages, methods for serializing objects between processes and across language boundaries are important. We have created a custom Lua wrapper to the MessagePack[8] encoder, so that we are able to send and receive arbitrary messages across the network quickly and reliably. Robot data are transformed to browser data: from Lua userdata to JavaScript typed arrays and from Lua tables to JavaScript objects. Also, the serialized data are easily logged and replayed, providing a way to diagnose and debug our processes.

For low-level access, the operator interface is an interactive Lua interpreter using the proven Secure Shell (SSH) protocol to manipulate state machines and shared memory. Additionally, on the operator console, reliably delivered TCP messages and unreliable raw UDP packets broadcast from the robot are bridged to WebSocket messages; any number of channels can be forwarded to the operator. This system relays only state machine events and shared memory access, similar to a command line interface; the main addition is listening for sensor data. The sensor feeds are toggled on and off and modified via shared memory variables.

While these interprocess channel assets reside on one single computer, we leverage request/reply methods and UDP fallbacks to provide a remote operator with access. For example, the remote operator can perform high-level arm behaviors by sending final-state machine (FSM) transition events, or even directly manipulate the joint angles by remotely accessing shared memory variables. For high bandwidth sensor feeds, the user can dynamically select between reliable and unreliable protocols.

We have tested extensively with settings on the Mini Maxwell network shaper of the competition. We have tested our network setup under more duress than the competition would provide, dropping 25% of packets, enabling reordered packets, and doubling the round trip lag to 2,000 ms. We are able to communicate effectively with the robot, and we observed no incorrect behavior. Our testing provided a larger degree of safety certainty and assurance of at least network robustness.

Shown in Figure 9 is an example of our network usage for a 2 min window in which THOR-OP was picking up the drill. Data spikes occurred when we requested a "slow mesh." Sometimes, the mesh requests clog our network under poor network settings. When this occurs, we turn off our camera feed to let the buffer of delayed packets empty, as was done around the 80 s mark. As can be seen, a minimal amount of data was sent to the robot.

### 3.5. Simulation Support

We use Webots (Michel, 2004) as the simulation environment for development. Webots is a multiplatform commercial robotic simulator we successfully have been using with

[8]Messagepack serialization library. http://www.msgpack.org

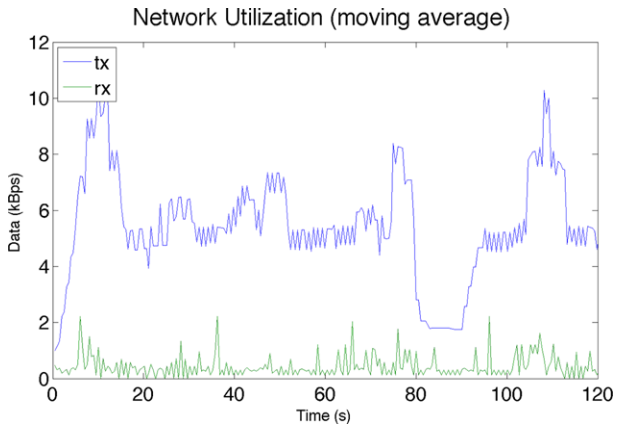



**Figure 9.** Network usage for picking of the drill during poor network conditions. A 5 s moving average was applied.

our previous development for RoboCup. It supports a variety of sensors and actuators out of the box, and the simulated sensors can be easily turned on and off depending on needs. On our laptop computers, we simulate the robot, with sensors turned on, three times faster than real time, or around 15 to 20 times faster than real time with the sensors off.

The mass model for the simulation is calculated from the CAD model of the THOR-OP robot, which we have found to be fairly accurate. Due to the modular construction, a large proportion of the total mass is concentrated at the actuators. Thus, our simulated model with point masses at the actuator positions closely matches the actual robot.

In addition, the ease of adding and modifying models in simulation helps us to design the passive end-effectors. We have improved and validated each design by running many manipulation tasks and checking performances in the simulation. This process saves a great deal of time and effort for prototyping and manufacturing.

## 4. UPPER BODY CONTROL

We have constructed an upper body motion controller that governs the neck, arms, and waist joints. Bimanual manipulation itself is a challenging task, especially when each arm has a redundant DOF. It becomes even harder with humanoid robots, as they should balance themselves against a changing mass distribution and external forces during manipulation. In this section, we describe how we have designed our upper body motion controller to satisfy these requirements.

### 4.1. Arm Controllers

At the lowest level, we apply direct control over each joint angle. This control mode is useful in emergency situations, for instance when the standard inverse-kinematics (IK) -based control is unable to generate solutions at singular points. As the full body balancing controller is suspended during the joint-level control, we limit its use for emergency situations only.

Another way to control the arms is by controlling the target pose of end-effectors in a 6D Cartesian space. The operator specifies the desired target pose of end-effectors by position $\{x, y, z\}$ and orientation in Euler angles $\{\varphi, \theta, \psi\}$, then the arm planner calculates, in joint angle space, the trajectories for the end-effector to reach the target pose.

Due to the 45 degree offset of the end-effector's palm, the change of the target orientation alone usually results in the large movement of the whole arm, which is not desirable. Thus, we also provide the wrist rotation control, which only changes the wrist joint angles to modify the end-effector orientation.

Some motions require additional constraints on the end-effector trajectories, so we provide a number of task-specific, parametrized arm trajectories and allow the operator to have control over the parameters. For example, the door opening motion has the following parameters:

$$P_{\text{open door}} = \left\{ xy_{\text{hinge}}, y_{\text{knob}}, z_{\text{knob}}, y_{\text{grip}}, \varphi_{\text{knob}}, \psi_{\text{door}} \right\}, \quad (1)$$

where $xy_{\text{hinge}}$ is the $x$ and $y$ coordinate of the door hinge, $y_{\text{knob}}$ and $z_{\text{knob}}$ are the $y$ displacement and the height of the door knob axle, $y_{\text{grip}}$ is the relative $y$ displacement of the gripping position from the door knob axle, and $\varphi_{\text{knob}}$ and $\psi_{\text{door}}$ are the roll and yaw angles of the door knob and the door. The operator can specify the knob angle parameter $\varphi_{\text{knob}}$ to rotate the knob and the door angle $\psi_{\text{door}}$ to open or close the door.

#### 4.1.1. Task Level Control

For the DRC Trials, we approached each manipulation task as a sequence of *phases*, e.g., gripping phase, opening phase, turning phase. During each phase, the operator was able to fine-tune the active controller, be it joint angles, end-effector poses, or parametrized trajectories. At the end of each phase, the operator can choose to advance to the next phase, fine-tune end-effector poses, or revert back to the previous phase. We make sure that all the phases are reversible, so that the robot can retry subtasks with minimal delay. Figure 10 shows the intermediate phases of the loaded door opening task.

### 4.2. Arm Planner

Except for the joint-level control, the arm control interface generates a sequence of target poses for the end-effectors as a reference trajectory. These poses are passed to the arm planner to calculate the corresponding sequence of joint angles. The robot must calculate a trajectory connecting the current pose through the target poses, conforming to the joint limits and kinematic constraints. We use a simple linear

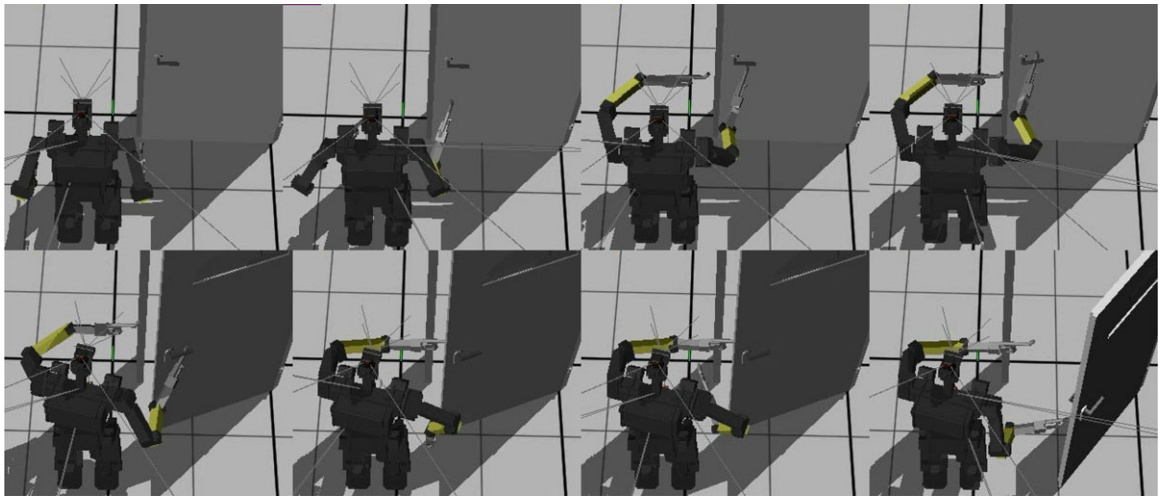


**Figure 10.** The multiple phases of the loaded door opening task.

interpolation over Cartesian space for trajectory generation. Due to the redundancy of the arm's DOF, we can find IK solutions for most of the points in the straight line between two poses.

#### 4.2.1. Joint Angle Trajectory Generation

For each pose in the pose trajectory, we calculate the corresponding joint angles to build a joint angle trajectory. The arm has redundant DOF yielding an infinite number of IK solutions, thereby requiring a costly optimization problem to find the best solution in general. To simplify the process, we keep only one joint angle, the shoulder yaw angle, as the variable to optimize. If we fix one joint angle, the arm has now only 6 DOFs with colocated wrist joints, which has an analytical inverse kinematics function that can be calculated efficiently. The resulting joint angles become an analytic function of the shoulder yaw angle, which we can minimize easily. We define the cost function as

$$\begin{aligned} J = & \sum_{i=1}^{7} [M(m_i, j_i(k) - j_i^{\max}) + M(m_i, j_i(k) - j_i^{\min})] \\ & + k_1 M(m_6, j_6(k)) \\ & + k_2 \sum_{i=1}^{7} |j_i(k) - j_i(k-1)|, \end{aligned} \tag{2}$$

where $j_i$ is the joint angle, $j_i^{\min}$ and $j_i^{\max}$ are the joint angle limits, $m_i$ is the minimum margin allowed for joint $i$, and $k_1$ and $k_2$ are weight parameters. $M$ is the non-negative margin function, which is defined as

$$M(m_{\max}, x) = \max(0, m_{\max} - |x|). \tag{3}$$

Then the cost function penalizes joint angles near the joint angle limit, the wrist roll angle near the singularity, and large movements from previous joint angles. Optimization is done by greedy stochastic descent, which samples a number of shoulder yaw angles around the current angle, selecting the one with minimum cost. We have found that this sampling-based approach generates joint angle trajectories that effectively use all seven DOFs of the arms while avoiding possible singularities.

#### 4.2.2. Joint-level Interpolation

After we calculate the joint angles for all the poses in the pose trajectory, we interpolate the resulting joint angle trajectory in the joint space to generate a continuous joint-level movement. We chose the interpolation in joint space rather than interpolation in pose space, as the latter requires more computation and has the risk of confronting singularities, which may result in getting stuck or reaching joint velocities higher than the actuator limit. On the other hand, the interpolation in joint space is always possible, and we can use the angular velocity constraints to determine each duration of movement so that the end-effector always follows the reference trajectory while satisfying the joint velocity limits. One disadvantage of this approach is that the resulting end-effector pose trajectory will not follow a straight line, but we have found that with the resolution we use for the pose trajectory, the resulting motion is smooth enough for practical purposes.

### 4.3. Torso Movement Compensation

Manipulation is harder with humanoid robots, as compared to wheeled robots. Due to their upright posture and small support area, humanoids are more susceptible to toppling over with the additional weight of grabbed items, or even the relocated mass of outstretched arms. Any arm

movement can change the COM position of the robot, which, in turn, can make the robot unstable without proper balancing control.

We handle this problem by moving the torso so that the overall center-of-mass position remains fixed during arm motion. However, this torso movement will change the end-effector positions as well. The arm configuration should be adjusted to compensate for the shifted torso position in order to achieve the desired end-effector pose. If we assume the torso shift at the discrete time $k$ to be $\mathrm{COM}_{\mathrm{shift}}(k)$, it should satisfy the following simultaneous equations:

$$\begin{aligned} q_{\mathrm{larm}}(k) &= \mathrm{IK}_{\mathrm{larm}}[p_{\mathrm{larm}}(k) - \mathrm{COM}_{\mathrm{shift}}(k)], \\ q_{\mathrm{rarm}}(k) &= \mathrm{IK}_{\mathrm{rarm}}[p_{rarm}(k) - \mathrm{COM}_{\mathrm{shift}}(k)], \\ \mathrm{COM}_{\mathrm{shift}}(k) &= \mathrm{COM}_{\mathrm{ub}}[q_{\mathrm{larm}}(k), q_{\mathrm{rarm}}(k)], \end{aligned} \tag{4}$$

where $q_{\mathrm{larm}}$ and $q_{\mathrm{rarm}}$ are joint angles for each arm, $\mathrm{IK}_{\mathrm{larm}}$ and $\mathrm{IK}_{\mathrm{rarm}}$ are inverse kinematics functions, and $\mathrm{COM}_{\mathrm{ub}}$ is the function that calculates the COM displacement of the upper body given arm joint angles. As Eqs. (4) are nonlinear simultaneous equations, we use the following online approximation instead:

$$\begin{aligned} \mathrm{COM}_{\mathrm{shift}}(0) &= 0, \\ q_{\mathrm{larm}}(k) &= \mathrm{IK}_{\mathrm{larm}}[p_{\mathrm{larm}}(k) - \mathrm{COM}_{\mathrm{shift}}(k-1)], \\ q_{\mathrm{rarm}}(k) &= \mathrm{IK}_{\mathrm{rarm}}[p_{\mathrm{rarm}}(k) - \mathrm{COM}_{\mathrm{shift}}(k-1)], \\ \mathrm{COM}_{\mathrm{shift}}(k) &= (1-\gamma)\cdot \mathrm{COM}_{\mathrm{shift}}(k-1) \\ &\quad + \gamma \cdot \mathrm{COM}_{\mathrm{ub}}(q_{\mathrm{larm}}(k), q_{\mathrm{rarm}}(k)), \end{aligned} \tag{5}$$

which incrementally moves the COM offset $\mathrm{COM}_{\mathrm{shift}}$ to satisfy Eqs. (4). We tune the mixing parameter $\gamma$ to speed up or slow down the convergence to the desired pose.

## 5. LOCOMOTION CONTROL

In this section, we describe the lower body motion controller, which controls leg joints to make the robot walk. During the task, the robot is supposed to walk over various types of terrain. We provide a number of control interfaces to suit different tasks. Bipedal locomotion in uncontrolled environments is one of the most difficult tasks of the whole challenge, and it has been an active research topic for humanoid robotics for years.

### 5.1. Walk Controller Interfaces

This basic option provides the capability to change the walk velocity of the robot in real time. At the end of each step, the robot calculates the next step position based on its current foot configuration, commanded walk velocity, and kinematic constraints. This type of control allows for direct teleoperation of the robot using a joystick, and the robot can quickly change its velocity to adapt to a dynamic environment. However, as the communication bandwidth is strictly limited, and none of the DRC Trials tasks had a dynamically changing environment, we had little reason to use this interface for the DRC Trials.

The other interface option for locomotion specifies the target pose the robot should move to, which is defined as $\{x, y, \psi\}$. When the target pose is set, the step planner generates a number of consecutive steps considering the walk velocity constraint and kinematic constraint. As this option plans for the future foot step positions in advance, this allows the use of a zero moment point (ZMP) preview-based walk controller. We used this combination during most of the DRC Trials.

Although most of the tasks assume a relatively flat surface, some tasks require the robot to move over surfaces with different heights or inclinations. This situation sometimes forces the robot to take a much larger stride than usual. Larger strides are undesirable and are undertaken only as needed since they can hamper stability. We handle such cases as special events only performed by request, and we provide a separate control interface for them.

### 5.2. Walk Controller

The walk controller generates the torso and feet trajectories that make the robot step to the target poses while keeping dynamic balance. To cope with different requirements for locomotion, we provide multiple walk controllers with different characteristics. In addition to providing multiple walk controllers, we also provide the ability to dynamically switch between them during locomotion (Yi et al., 2013), which we name our hybrid walk controller.

The reactive walk controller (Yi et al., 2011b) is based on the analytic solution of the linear inverted pendulum model (LIPM) dynamics equation with fixed height. To get the solution in a closed form, we specified the reference ZMP trajectory as a piecewise linear function and put constraints on the boundary conditions of the COM. The resulting torso trajectory has velocity discontinuities at step transitions, which makes the robot stumble when it takes a first step. To solve this issue, we use the hybrid walk controller approach, where the first and last steps are handled by the preview walk controller.

The preview walk controller is based on the optimization of the torso trajectory to minimize the ZMP error over its preview period (Kajita, Kanehiro, Kaneko, Fujiwara, and Yokoi, 2003). By definition, it requires a ZMP trajectory in advance, so future foot step positions should be provided to calculate the trajectory. We assume a LIPM model and fixed center-of-mass height for the robot, which allows us to utilize an efficient quadratic optimization method, where the optimization can be iteratively done by a simple matrix multiplication at every step.

Finally, for more challenging terrains, the robot may need to take larger strides. Here, the point mass assumption of the LIPM can induce large amounts of error and make the

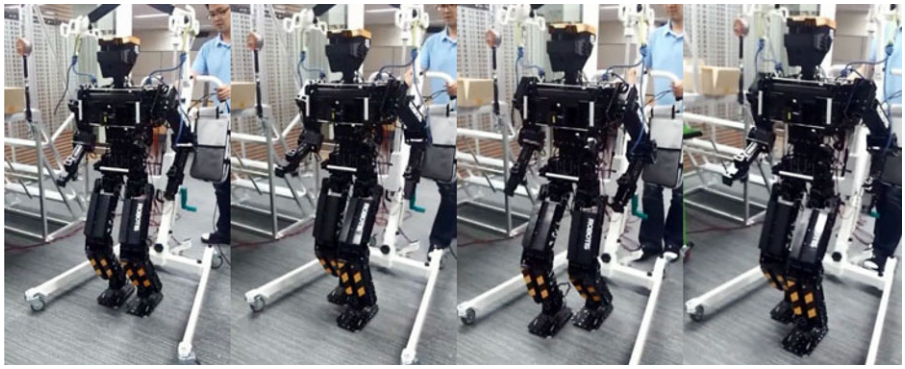


**Figure 11.** THOR-OP robot demonstrating dynamic bipedal walking capability.

robot unstable. For these steps, we have a more refined version of the preview walk controller, which relaxes the point mass assumption and fixed COM height assumption we use for other controllers. The initial torso trajectory is generated by the standard preview walk controller, and the ZMP error of the initially generated motion is calculated using a multibody model of the robot. Another stage of preview control is applied to minimize the multibody ZMP error. Although it generates the most stable trajectory among walk controllers, this controller requires much more computation than others, so we use this controller only for special steps. Figure 11 shows the robot in a walking experiment.

## 5.3. Balancing Controller

The robot has to withstand a number of static and dynamic perturbations from many sources, which includes uneven terrain, contact with surroundings, reaction forces from manipulation, and changes of mass distribution. We use various balancing controllers to stabilize the robot during the task.

### 5.3.1. Arm Movement Compensation

A dynamic movement of the arms, or even a different posture of arms, can negatively affect the balance of the robot unless it is compensated properly. The effect of the arm movement can be calculated in advance using the mass model of the robot in the ZMP of the upper body,

$$p_{\text{upper body}} = \frac{\sum\{m_i[(\ddot{z}_i + g)x_i - \ddot{x}_i z_i] + I_i\ddot{\theta}_i\}}{\sum m_i(\ddot{z}_i + g)}, \qquad (6)$$

where $m_i$ is the mass, $x_i$ and $z_i$ are the $x$ and $z$ positions of the center of mass, $I_i$ is the inertia, $\theta_i$ is the angle of the $i$th rigid body of the robot, and $g$ is the gravitational constant. The upper body ZMP $p_{\text{upper body}}$ can be subtracted from the reference ZMP to make the whole robot balanced during dynamic upper body movements.

During the DRC Trials, we used a conservative arm joint velocity limit. For quasistatic cases, we can ignore the linear and angular accelerations $\ddot{x}_i, \ddot{z}_i, \ddot{\theta}_i$ to simplify Eq. (6) as

$$p_{\text{upper body}} = \frac{\sum m_i g x_i}{\sum m_i g}, \qquad (7)$$

which is the $x$ component of the center of mass. Figure 12 shows the change of torso position to compensate for different arm configurations of the robot.

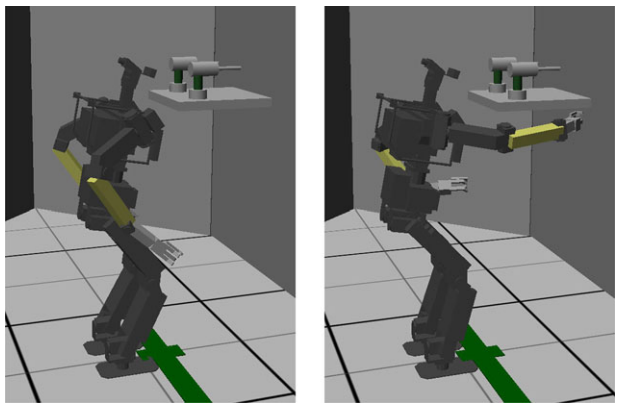
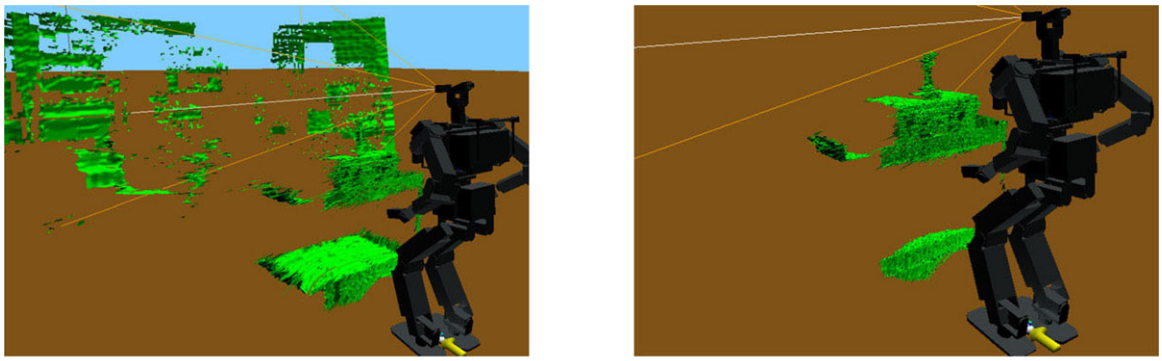


**Figure 12.** The quasistatic full body balancing control moved the torso significantly to offset the weight of the arms.

### 5.3.2. Push Recovery Control

Humans are known to perform a number of distinct behaviors to reject external perturbations. This includes the *ankle strategy*, which uses the ankle control torque to keep the center of mass within the support polygon, the *hip strategy*, which uses angular acceleration of the torso and free limbs, and the *step strategy*, which uses reactive stepping toward the direction of the external force to change the support polygon of the robot.

All three push recovery behaviors have been successfully implemented on position-controlled humanoid robots before (Yi et al., 2011a). We chose not to use the hip and



(a) (b)

**Figure 13.** 3D constructions provided from the LIDAR with different settings. (a) Fast mesh, (b) slow mesh.

step strategies at the time of the DRC Trials, as we were not sure of the long-term effect of the quick torso rotation on the actuators, and the preview-based walk controller did not allow for reactive stepping. Instead of those two push recovery strategies, we used a simple stop reflex in addition to the ankle strategy. This makes the robot stop walking and lowers the center of mass to resist large perturbations. The decision boundary to trigger a stop reflex can be determined by inspecting the theoretical stability region of the ankle strategy,

$$\left|\frac{\dot{x}}{\omega} + x\right| < \frac{\tau_{\text{ankle}}^{\max}}{\text{mg}}, \tag{8}$$

where $\omega$ is the natural frequency of the robot, and $\tau_{\text{ankle}}^{\max}$ is the effective maximum ankle torque, which can be found by trial and error for the actual robot.

Although simple, we have found that our push recovery controller works quite well in practice, which enabled THOR-OP to successfully survive many hazards, including inclinations, bumps, and multiple contacts.

## 6. PERCEPTION SYSTEM

The perception system is responsible for providing the human operator the information of the task environment, as well as the current and estimated states of the robot needed for motion control. Our sensor selection includes an IMU, multiple RGB cameras at the head and wrists, two LIDARs, and joint encoders in every joint. Among those sensors, the IMU and encoders are used constantly for state estimation and balancing of the robot; all others provide on-demand data for the operator, depending on need and network quality.

### 6.1. 3D Reconstruction

We solely use the LIDAR located within the chest of the robot, which scans vertically while being actuated horizontally. We use 90 degree and 60 degree vertical and horizontal fields of view, respectively, and store scans into a cache. On the operator computers, three-dimensional information is computed from the LIDAR depth cache based on mesh triangulation algorithms (Holz & Behnke, 2013).

To adapt to the limits of bandwidth usage, we filter and compress the LIDAR readings before sending to the operator. First, we filter depth information, saturating readings outside of user-adjustable bounds. The depth readings are then mapped into integers between 0 (lower bound) and 255 (upper bound). The integer values are compressed with either PNG or JPEG compression. Since the user can tweak the range filter and compression technique, we always keep the raw readings in memory.

When the robot is far away from the object to be manipulated, we use the "fast mesh" setting with bounds of 1 cm and 5 m away and lossy JPEG compression. This produces a noisy mesh for approximate navigation and distance estimation. Since the packet size is small, we can request this image quite frequently without incurring network penalties. When the robot is close to an object of interest, the operator obtains a "slow mesh" that uses bounds of 1 cm and 1 m with lossless PNG compression. This finer resolution cannot be requested often under poor network conditions, however. Figure 13 shows 3D construction results with both settings.

### 6.2. Video Processing

We used multiple camera streams to provide robot views to the human operator. The main camera was mounted at the head, with two additional cameras mounted on each hand. The operator views from the hands during manipulation proved extremely helpful for turning the valve and gripping the drill, which required precise depth perspective at all times. Figure 14 shows the camera images from different perspectives.

The quality of both the head and hand cameras can be specified by the user through shared memory variables.

(a) (b)

**Figure 14.** Multiple cameras provided different perspectives for manipulations. (a) Head camera, (b) hand camera.

**Table III.** Sensor stream settings.

| Visual | Compression Format | Quality (0–100) | Interval (Hz) | Frame Size (kB) |
|---|---|---|---|---|
| Fast Mesh | JPEG | 90 | 0–0.5 | 5–20 |
| Slow Mesh | PNG | – | – | 30–40 |
| Head Image | JPEG | 60 | 0.5–2 | 3–5 |
| Hand Image | JPEG | 50 | 0.5 | 3–5 |

Resolutions are 160 by 120 pixels for the hand camera and 320 by 180 pixels for the head camera typically. In general, we modulate the transmission rate between 0.5 and 5 Hz, with JPEG compression primarily. Certain situations (gripping the drill, for instance) call for a single high-quality PNG image. In general, low resolution and low frequency provide adequate operator awareness of the environment and end-effector positioning during the competition. We have found that at higher resolutions (large amounts of data), with the poor network profile, packets begin to overflow the network, and we incur latencies of around 10 s until the good network profile is activated. Due to this, we kept our camera settings conservative through the competition, as shown in Table III.

### 6.3. Audio Feedback

One issue we have had with testing the DRC tasks is that it is can be hard for the operator to determine whether the robot has successfully triggered the drill based on low-frame-rate video alone. To handle this issue, we use the built-in microphone of the head camera to get audio feedback from the robot. The remote operator, when needed, requests a 5 s audio clip recorded at 16 kHz, compressed in the MP3 format with a bit rate of 8 kbps. The size of such an audio file is under 30 KB (similar to our PNG mesh images), so it adds little burden to the network when being transmitted.

**Figure 15.** The operator interface setup included three main screens for the user to guide the robot and observe its environment.

## 7. OPERATOR CONSOLE

The basic operator setup is shown in Figure 15, which includes a laptop, an external display, and a tablet. The monitor displays the head camera feed, while the tablet takes touch inputs for gripper control and displays the visual feedback or the hand camera feed. The main laptop screen shows the 3D scene with the robot model and the pertinent buttons for commanding the robot. During the DRC Trials, we used a second laptop to monitor various processes and provide low level robot control. Figure 16 describes the system layout of our operator console that allows for multiple machines to be used simultaneously.

We access shared memory and send state machine events through a Lua interpreter over SSH for low-level modifications of robot behavior, with hotkey scripts for common tasks. Figure 13 shows a complementary in-browser graphical user interface (GUI) that, using the

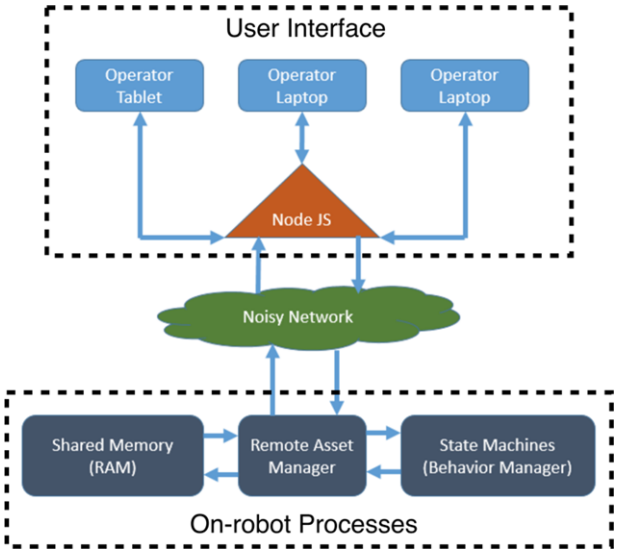



**Figure 16.** The system layout for the operator interface allowed multiple machines to be used simultaneously.

THREE.js[9] framework, shows a mesh of LIDAR returns and a figure of the robot from joint and inertial feedback.

The human operator interacts with the GUI in a number of ways. Standard hotkeys request LIDAR data, send state machine events, or modify object parameters. Double clicking on regions of the 3D scene perform raycasting to "pick" points in the virtual world and trigger special callback functions. As we have described in Section IV, the task level manipulation is composed of a sequence of parametrized motions with subtask-specific model parameters such as Eq. (1). The operator workflow revolves around acquiring and fine-tuning these model parameters.

### 7.1. Model Matching

We developed autonomous algorithms to determine the model parameters for certain tasks. Our algorithms smooth the 3D mesh and extract contour lines before optimizing in parameter space to find the best fit. The algorithm is performed at the operator side, and we found it to work well for tasks such as the valves or walls. However, for the actual DRC Trials, we decided to use the slower but more reliable human-in-loop approach.

We let the operator analyze the current sensory information and visually match the model parameter using the GUI. The operator is given multiple camera video feeds and the 3D mesh of the manipulation target. Once the initial model is acquired through the GUI, we overlay object-specific 3D geometries over the scanned mesh, such as a torus for the valve and a cylinder for the hose. The operator can further move and rotate the visualized model to fine tune and match the 3D mesh. We have found this process to be intuitive and efficient, and it took only a few seconds for the operator to acquire the model parameters in most cases. For us, avoiding errors is favored over quick task execution, but fast and reliable autonomous humanoid manipulation remains a challenging area of research.

[9]R. Cabello. Three.js JavaScript webgl library. http://www.threejs.org

### 7.2. Model Override Control

If the robot is given the precise model parameters for the current subtask, the robot should be able to autonomously complete the manipulation subtask without any human intervention. In reality, we have found that sometimes the initially perceived model is not good enough for continual usage, and we need to manually update it based on visual feedback. We prefer fine-grained control over our robot's behavior for many tasks. For example, the robot is supposed to fully close the valve until the steam flow ends: it would be disastrous if the robot had to rotate the valve a few more degrees yet the user could not send such a command. Therefore, we provide the control interface to modify the task-specific model parameters such as Eq. (1) in real time during manipulation.

In emergency cases, the operator can directly move the end-effectors outside of the model constraints, by specifying the target position $\{x, y, z\}$ and orientation in Euler angles $\{\varphi, \theta, \psi\}$. During the Trials, we had no critical emergency cases in which we used these low-level end-effector overrides.

### 7.3. Gripper Interaction

We devote an entire tablet for manipulation control. Through another web page, the user is able to request position, temperature, and other diagnostics from the gripper servo motors. Additionally, the user can execute a few grip commands for each finger. Since we use torque control (via desired current) for the fingers, we allowed for high (900 mA), low (200 mA), zero, and opposing (45 mA) currents for triggering, gripping, softening, and opening, respectively.

### 7.4. Locomotion Interface

We provide a simple locomotion control driven by waypoints in either the local frame or the global frame. As noisy odometry information is our sole source of localization, we ensure that we never command the robot to move too far at one time. When the manipulation target is far away, we let the operator manually select intermediate target poses. Although our walk controller supports full omnidirectional walking, we prefer to give only one direction at a time—forward, sidestep, or turn—to maximize the stability during the locomotion.

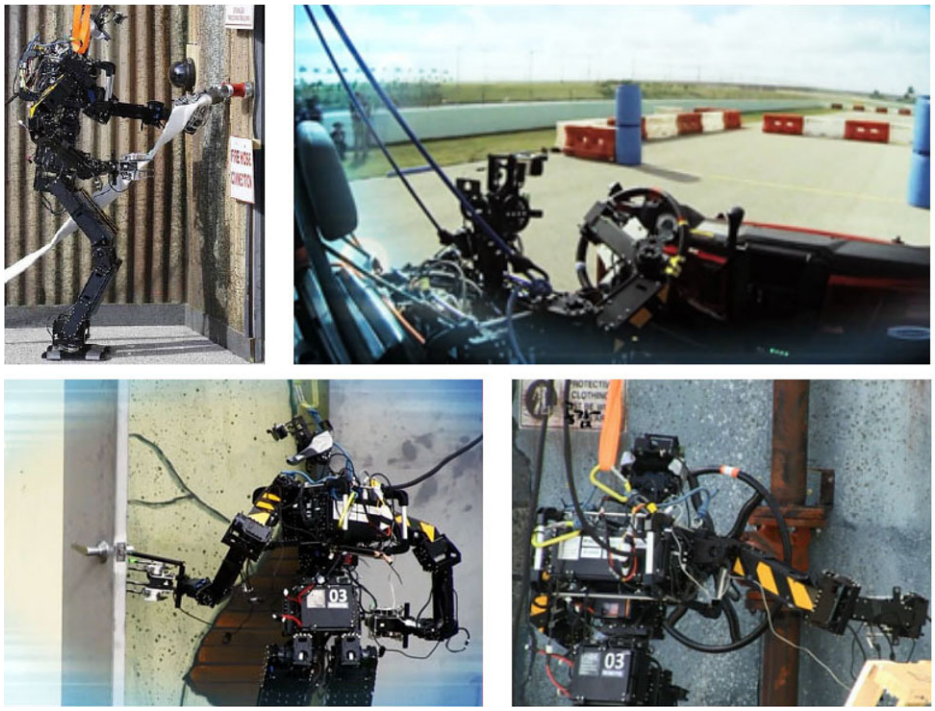


**Figure 17.** THOR-OP at the 2013 DRC Trials performing four of five tasks in which it scored points.

Using the task-specific model parameters, we guide the robot to the ideal position for manipulating the target object. The ideal position, stored as a local offset from the object, is sent to the footstep generator to calculate a number of omnidirectional footsteps. As our footstep generation algorithm assumes free space without obstacles, we only use this control for the fine-tuning positioning for manipulation targets.

## 8. TESTING AND TRIAL PERFORMANCE

The DRC Trials 2013 were held at the Homestead Motor Speedway in Florida. THOR-OP, representing Team THOR, attempted all eight tasks set forth by the DRC and competed well, only succumbing to a couple of unexpected hardware and setup logistics issues. Figure 17 shows THOR-OP performing several tasks during the competition. Overall, we accrued eight points and finished in 9th place out of 16 teams (see Table IV). In this section, we would like to present in detail the experiments during preparation as well as the trial performances for some of the tasks.

### 8.1. Vehicle Task

Only 7 out of 16 teams attempted this task, and eventually four teams scored. Although our team had never practiced for this task before the trials, we ended up scoring one point, which was a big achievement as no team could get the second point. We confidently attempted this task due to several considerations. First, THOR-OP, being relatively small and lightweight, was easy to accommodate into the vehicle without many limitations on the robot's workspace. Another reason was that the driving task conceptually consisted of two simple tasks: turning the steering wheel and stepping on/off the throttle. The former was similar to the valve task, about which we were quite confident, and stepping on and off the throttle basically only involved the control of the ankle and knee pitch motors, which could be performed easily with direct joint-level control.

A passive end-effector consisting of three rods in a triangular pattern was mounted on the robot's right hand for steering the wheel. Markers were taped on the steering wheel to help the operator visually see how much the wheel had rotated. In addition to the head camera providing the front view, another camera on the left gripper was used to monitor the vehicle's front left wheel to increase the certainty in steering control. The vehicle's acceleration was adjusted by changing the duration of the robot's foot pressing down on the throttle. We toggled between 1.5 and 3 s depending on various situations.

Team THOR completed the driving in about 22 min, without hitting any obstacles. It took 2–2.5 min for THOR-OP to pass each barrel except for the first one, which took about 8 min because the operator needed time to get familiar with the course and the operations. A simple stop-steer-move strategy was applied and the route the vehicle took was quite smooth without hard turns.

**Table IV.** Trial performance of team THOR.

| | Vehicle | Terrain | Ladder | Debris | Door | Wall | Valve | Hose |
|---|---|---|---|---|---|---|---|---|
| Scores (max. 4) | 1 | 0 | 1 | 0 | 1 | 0 | 4 | 1 |
| Interventions | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| Time Duration | 22:27 | 30:00 | 07:14 | 06:24 | 29:45 | 30:00 | 15:31 | 30:00 |

### 8.2. Valve Task

Instead of using an actuated gripper, we decided to use a passive two-spoke mechanism as the end-effector for the valve task in pursuit of reliability and efficiency. The robot simply needed to align the end-effector to the valve center and maintain its perpendicularity to the valve plane. Then the valve turning task could be performed by rotating only the wrist, rather than moving the whole arm. It largely simplified the motion control and reduced the power consumption for the valve task from about 140 to 100 W. Moreover, using a passive end-effector allowed continuous rotation of the wrist without cables being tangled.

The use of the hand camera on the other gripper to feature a side view of our valve approach played a crucial role in improving our success rate on this task by increasing the operator's awareness of the relative position of the end-effector to the valve.

During our official trial, the most time-consuming part turned out to be the robot's approach to the desired position. Here, the workspace of the arm included the task-specific workspace needed to turn the valve entirely. To ensure a stable and less drifted walk during the competition, the robot was only commanded to move in pure rotation or pure translation in one direction. One unexpected situation in the valve task was that a high torque was required to turn the small valve, yet the full body stability and two-spoke passive mechanism enabled us to complete the rotation without a problem. Team THOR was the fastest to complete the valve task without any intervention, using less than 16 min, and it was awarded Best Task in Valve.

### 8.3. Door Task

We performed ample tests, both simulation and the real world, to determine the most suitable body pose and approach strategy for this task. Although THOR-OP can technically walk forward through a door frame without touching it, we found it too hard to do, especially without good situational awareness. Instead, side steps were chosen for walking through the doors. We first tested with the robot initially facing the door so that we could obtain a 3D representation from the chest LIDAR. However, it took too much time and effort for the robot to rotate by 90 degrees and align to the center of the door frame. We eventually decided to let the robot start sideways with respect to the door, and slowly yet steadily take side steps to pass through the door. The cameras on the hand and head were our primary sensors. The hook, another passive end-effector that extended from the gripper, was used to easily maneuver the door handles without any stress to the gripper actuators.

**Figure 18.** THOR-OP used a hook to hold and rotate the handle for the door task. Almost 80 degrees was required to open the door, which was unexpected and harder than our testing setup, which only required 60 degrees.

The locomotion of our robot was quite stable during the door task competition, in spite of many contacts with the environment due to limited situational awareness. Walking toward the first door, getting through it, and approaching the second door took 2 m, 54 s; 4 m, 14 s; and 4 m, 15 s; respectively; these were even better than our testing performance. It took nearly 8 min to open the first door, which was much longer than expected due to the nearly 80 degree angle as opposed to the proposed 60 degree angle needed to rotate the handle for opening the door, as shown in Figure 18. Being aware of this, the operator applied more rotation on the handle for the second door and used just 3 min to open it.

Unfortunately, there were strong winds on the first day of the DRC Trials, which impeded our progress but also portrayed our full body balancing and stable walk. The THOR-OP had to open the second door three times since the wind kept pulling back and shutting the door. Since we used a passive mechanism, we avoided any damage to our gripper that might have occurred from the door hitting the end-effector multiple times.

### 8.4. Wall Task

The most difficult part we found in preparing for the wall task was to have a good grasp of the drill. Since the gripper was designed specifically for the DeWalt drill used in the DRC Trials with a contour perfectly fitting the drill, we had to precisely direct it to a good position for grasping. Failing to do so resulted in an undesired grasp of the power tool and an increase in the load on the fingers. It required lots of practice for the operator to realize the optimal approach of the body and gripper to pick up the drill. Therefore, we added fiducial markers on the 3D representation. We also utilized audio feedback to relay back to the human operator that the tool was triggered.

During the DRC Trials, we successfully approached the table with the drills on it, and we carefully lined up and gripped the drill successfully, as is seen in Figure 19. As expected, 7 min was spent on the approach and fine adjustment of the body pose. After the first attempt at the drill, our operator decided to reapproach for a better body position. Therefore, a second sequence of walking commands was executed with the arm withdrawn. Another 4 min was spent aligning the gripper to a good position for holding and triggering, which was faster than the average performance from our testing.

However, as the robot started moving toward the wall, the rubber sole under its feet started to peel off, resulting in unstable walking. This did not happen in our practice since the floor where we tested had a much smoother surface than the one at the trials. This incident confirmed our concern that the glued rubber was not an ideal solution for more stable and reliable walking. Also, the belay on top of the robot was too short to allow for comfortable side stepping, and this made the robot fall when it tried to approach the wall to cut the predefined triangular path on the half-inch dry wall.

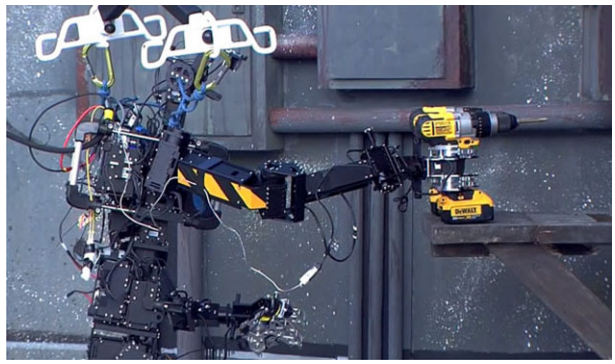


**Figure 19.** THOR-OP successfully approached to the table and got grasp of the drill. The upper finger was actually pressing the trigger of the drill at the moment shown in the figure.

### 8.5. Hose Task

For the hose task, the robot needed to perform three sequential tasks: pick up the hose end, drag and lift it to touch the wye, and screw the hose end to the wye thread. There were two main challenges in this task: to drag the hose without losing balance while walking, and to align the hose end perfectly to the wye thread in order to screw it in.

During experimental tests, we found that pulling the hose along the side of the robot affected the robot walking much more than pulling from the back. To handle this problem, we made a specific arm posture that placed the hand with the hose behind the robot close to the robot's center of mass, so that the robot was always pulled from behind.

At the DRC Trials, THOR-OP was able to deliver a stable and reliable walk while holding the hose. Unfortunately, just a few centimeters away from scoring our second point, the power cable to the left wrist actuator became unplugged. This disconnect meant the gripper went limp and out of position, precluding the robot from touching the wye with the hose, shown in Figure 20. From inspection, the cable got caught on the forearm when the gripper performed a large amount of rotation. We have henceforth replaced those cables with longer ones and reduced risky maneuvers in arm movements. It is important to stress that the clever four-bar linkage mechanism of the gripper prevented the hose from falling out even when there was no power to the fingers.

### 8.6. Debris Task

The debris task required the robot to clear a hallway by removing a number of wooden pieces. One of the main differences between the debris task and other manipulation tasks was that the locations of the target objects were low

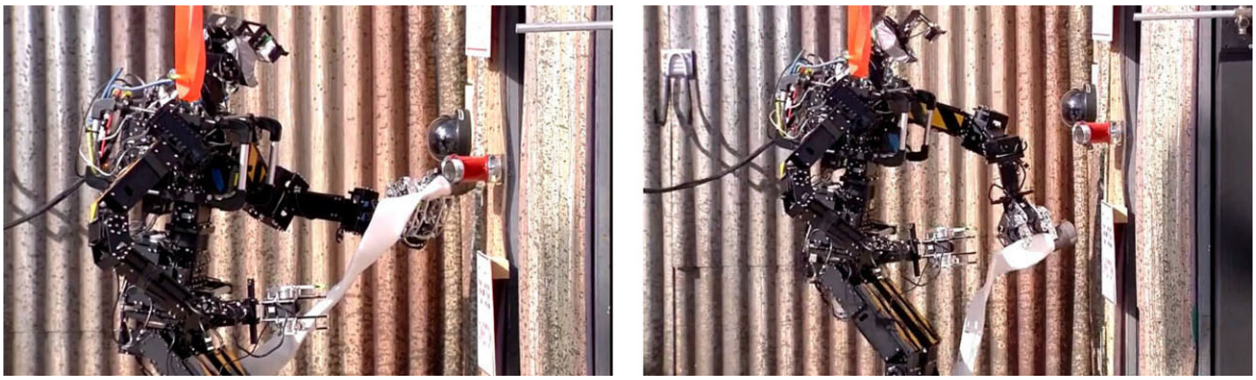


**Figure 20.** THOR-OP was almost there to touch the wye with the hose head and score the second point when the power cable for the left wrist popped out. The forearm drooped down but the gripper was able to keep a good grip of the hose.

with respect to the robot, so in order for the gripper to reach the object, we could not use the default standing posture as we used for other tasks.

We experimented with a variety of ways for the robot to reach down further, including kneeling down fully, lowering the body height by bending the knees more, and changing the torso pitch angle. We found that fully kneeling down or extensive knee bending put too much load on the knee actuators, and the robot could not walk normally with such a posture. On the other hand, we could increase the pelvis angle and the waist pitch angle to enable the robot to reach all the objects, while maintaining the locomotion capability.

During the preparation before we set out to perform the debris task, we tried to make the robot bend down to pick up a piece of debris five inches above the ground. This motion jammed one of the power connectors in the torso and shorted it out, causing the robot to collapse. We had to withdraw from this task due to this hardware issue.

### 8.7. Terrain Task

Our goal for the terrain task was to score one point, which requires traversing the pitch ramps and chevron hurdle of cinder blocks. During testing, we could reliably score the initial point in time. It was crucial for THOR-OP to precisely align the feet at proper distances to the ramp ridge or the edge of the hurdle, so that the robot would not fall or kick the cinder block. This process took most of the time spent for the task.

On competition day, one of the actuators on the left leg was broken during a test run just a few minutes before the terrain task. We managed to replace it but no time was left to bias the actuator, which affected the locomotion stability and reliability. THOR-OP successfully traversed the ramp, but an intervention was called since the robot lost balance when taking a huge step over the ramp. After the intervention, the operator was more cautious and spent more time,

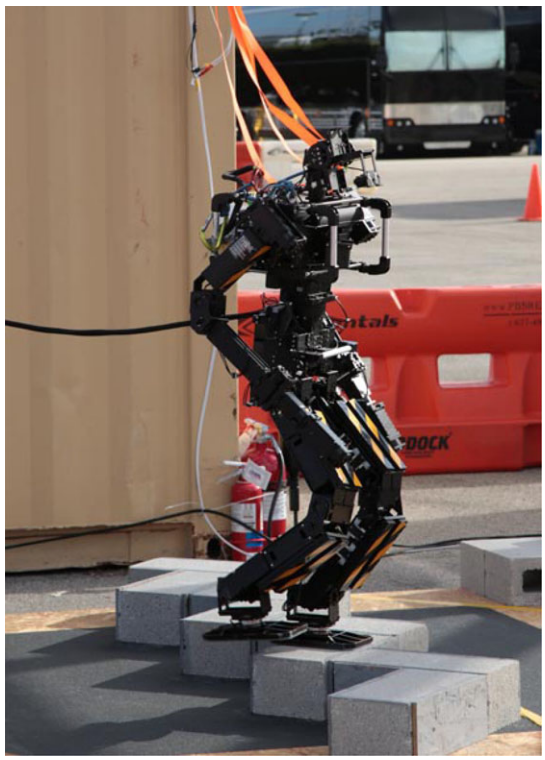


**Figure 21.** THOR-OP taking a step on the cinder blocks.

nearly 10 min, on positioning the robot to ensure a good position for stepping on and off the hurdle. THOP-OP succeeded in stepping onto the hurdle, as depicted by Figure 21, but time ran out before the robot stepped off.

### 8.8. Ladder Task

To obtain at least one point in the ladder task, the robot needed to climb onto the first rung of the ladder and all

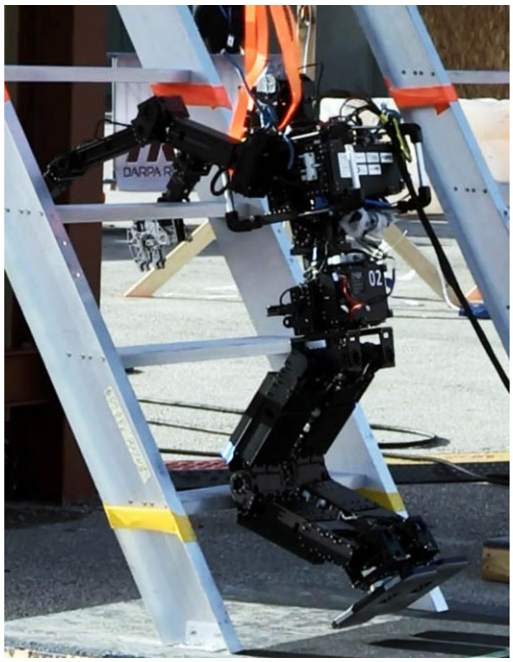


**Figure 22.** THOR-OP successfully scored a point in the ladder task.

points of the robot body must be off the ground. Due to the limited development time and hardware capability, we barely tested the ladder task before the DRC Trials. We still attempted the task and scored one point.

The camera on the head was used for perceiving the environment and monitoring the locomotion status. The THOR-OP approached the ladder and clung to the third rung with both arms, leaving the whole body leaning against the ladder. With both knees supported by the first rung and the gripper holding the ladder tightly, the robot then lifted both feet off the ground and scored, shown in Figure 22.

## 9. SUMMARY

THOR-OP has a number of distinct features from other robots in tracks A and B—it is fully modular, lightweight, and affordable. These attributes are desirable for immediate disaster response. During the DRC Trials, we proved that the THOR-OP robot, although being lightweight and compact, is capable of many practical disaster-response tasks.

For the software, we preferred a simple, conservative, and human-in-the-loop approach for a number of reasons. Our hardware was still at the prototype stage, and we had a tight development schedule and limited robot hours before the competition. Also, as we were allocated a long time for each task (30 min), the cost of possible failure greatly outweighed any performance gain we might achieve. For the upcoming DRC Finals, we may have a much shorter time to complete each task. This will emphasize more autonomous behavior of the robot, more optimized motion planning, and utilizing the maximum performance of the hardware.

Our remote operator console provided robust accommodation of the poor network, but we did not take advantage of good network situations as much as we could have. Being able to adapt to network conditions autonomously would help to give the user a more fluid experience in the best cases, possibly allowing for significant performance gains. THOR-OP is officially a finalist for the DRC Finals, and we hope that the rich experience from the DRC Trials will enable us to perform well. Having established a capable hardware and software platform, we are preparing to improve and test for the finals.

This article has provided a detailed description of Team THOR's algorithms and technical approaches to the 2013 DARPA Robotics Challenge Trials. To handle the great challenge of developing a bipedal disaster-response robot from scratch, we focused heavily on modularity of both hardware and software structures. Important benefits included the rapid field repairability of the robot, as well as the low development and manufacturing costs—all vital aspects for any robotic approach to disaster response. The DRC Trials results show that our hardware and software comprise a capable platform. Our future work will focus on providing more robot autonomy, incorporating the full dynamic properties of the robot for motion planning and balancing, and adding more analysis of high-dimensional sensor feeds.

## ACKNOWLEDGMENTS

We acknowledge the Defense Advanced Research Projects Agency (DARPA) through grant N65236-12-1-1002. We acknowledge the support of the ONR SAFFIR program under contract N00014-11-1-0074. We would like to thank DARPA for the photographs of Trials. We would also like to thank Taylor Presek from Virginia Tech for sharing some of the photographs of our robot from the Trials.

## REFERENCES

Butzke, J., Daniilidis, K., Kushleyev, A., Lee, D. D., Likhachev, M., Phillips, C., & Phillips, M. (2012). The University of Pennsylvania Magic 2010 multi-robot unmanned vehicle system. Journal of Field Robotics, 29(5), 745–761.

Collobert, R., Kavukcuoglu, K., & Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. In BigLearn, NIPS Workshop.

Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., & Maisonnier, B.

(2008). The nao humanoid: A combination of performance and affordability. CoRR, abs/0807.3223.

Han, J. (2012). Bipedal walking for a full-sized humanoid robot utilizing sinusoidal feet trajectories and its energy consumption. Ph.D. thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Holz, D., & Behnke, S. (2013). Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. In Intelligent autonomous systems (vol. 12, pp. 61–73). Springer.

Ishida, T. (2004). Development of a small biped entertainment robot qrio. In Micro-nanomechatronics and human science, 2004 and The fourth symposium micro-nanomechatronics for information-based society, 2004. Proceedings of the 2004 International Symposium (pp. 23–28).

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., & Yokoi, K. H. K. (2003). Biped walking pattern generation by using preview control of zero-moment point. In Proceedings of the IEEE International Conference on Robotics and Automation (pp. 1620–1626).

Kaneko, K., Kanehiro, F., Kajita, S., Hirukawa, H., Kawasaki, T., Hirata, M., Akachi, K., & Isozumi, T. (2004). Humanoid robot hrp-2. In Robotics and Automation. Proceedings, ICRA '04. IEEE International Conference (vol. 2, pp. 1083–1090).

Kim, J., Lee, Y., Kwon, S., Seo, K., Kwak, H., Heekuk, Lee, & Roh, K. (2012). Development of the lower limbs for a humanoid robot. In Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference (pp. 4000–4005).

Laliberte, T., Birglen, L., & Gosselin, C. (2002). Underactuation in robotic grasping hands. Machine Intelligence and Robotic Control, 4(3), 1–11.

Lim, B., Lee, J., Kim, J., Lee, M., Kwak, H., Kwon, S., Lee, H., Kwon, W., & Roh, K. (2012). Optimal gait primitives for dynamic bipedal locomotion. In Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference (pp. 4013–4018).

McGill, S. G., Brindza, J., Yi, S.-J., & Lee, D. D. (2010). Unified humanoid robotics software platform. In The 5th Workshop on Humanoid Soccer Robots.

Michel, O. (2004). Webots: Professional mobile robot simulation. Journal of Advanced Robotics Systems, 1(1), 39–42.

Muecke, K., & Hong, D. (2007). Darwin's evolution: Development of a humanoid robot. In Intelligent Robots and Systems, IROS. IEEE/RSJ International Conference (pp. 2574–2575).

Ogura, Y., Aikawa, H., Shimomura, K., Morishima, A., Lim, H.-O., & Takanishi, A. (2006). Development of a new humanoid robot wabian-2. In Robotics and Automation, ICRA. Proceedings of the IEEE International Conference (pp. 76–81).

Park, I.-W., Kim, J.-Y., Lee, J., & Oh, J.-H. (2007). Mechanical design of the humanoid robot platform, hubo. Advanced Robotics, 21(11), 1305–1322.

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

Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N., & Fujimura, K. (2002). The intelligent asimo: System overview and integration. In Intelligent Robots and Systems. IEEE/RSJ International Conference (vol. 3, pp. 2478–2483).

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

Yi, S.-J., Zhang, B.-T., Hong, D., & Lee, D. D. (2011a). Online learning of a full body push recovery controller for omnidirectional walking. In IEEE-RAS International Conference on Humanoid Robots (pp. 1–6).

Yi, S.-J., Zhang, B.-T., Hong, D., & Lee, D. D. (2011b). Practical bipedal walking control on uneven terrain using surface learning and push recovery. In International Conference on Intelligent Robots and Systems (pp. 3963–3968).