2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

Modular Low-Cost Humanoid Platform for Disaster Response

Seung-Joon Yi*, Stephen McGill*, Larry Vadakedathu*, Qin He*,
Inyong Ha†, Michael Rouleau‡, Dennis Hong‡† and Daniel D. Lee*

***Abstract*— Developing a reliable humanoid robot that operates in uncharted real-world environments is a huge challenge for both hardware and software. Commensurate with the technology hurdles, the amount of time and money required can also be prohibitive barriers. This paper describes Team THOR's approach to overcoming such barriers for the 2013 DARPA Robotics Challenge (DRC) Trials. We focused on forming modular components – in both hardware and software – to allow for efficient and cost effective parallel development. The robotic hardware consists of standardized and general purpose actuators and structural components. These allowed us to successfully build the robot from scratch in a very short development period, modify configurations easily and perform quick field repair. Our modular software framework consists of a hybrid locomotion controller, a hierarchical arm controller and a platform-independent operator interface. These modules helped us to keep up with hardware changes easily and to have multiple control options to suit various situations. We validated our approach at the DRC Trials where we fared very well against robots many times more expensive.**

Keywords: DARPA Robotic Challenge, Humanoid Robot, Modular Design, Full Body Balancing Controller

I. Introduction

The ultimate goal of humanoid robotics is to work in typical environments designed for humans. However, until very recently, most humanoid robotics research focused on small subsets of that goal – such as walk control, human robot interaction or stationary manipulation in well controlled laboratory environments. Attempts to build a complete, robust, remotely operated humanoid robot system for practical tasks have been deterred mainly due to the steep hardware costs as well as the risks associated with operating an intrinsically unstable humanoid in uncontrolled environments.

Sponsored by the United States Department of Defense, the DARPA Robotics Challenge [1] (DRC) pushes for big breakthroughs in humanoid robotics by providing a standard platform humanoid robot [2] and funding to selected teams, as well as special prizes for the winning team. The DRC specifically requires a complete system that can use human tools, drive a vehicle, and move around in unstructured human environments. Laden with uneven blocks, ladders, doorways, many such tasks require full body control and cohesion amongst the software components hitherto not seen before. Such requirements pose a great challenge for hardware and software; an especially short preparation time for the DRC Trials further mandates a very limited development period allowed for each team.

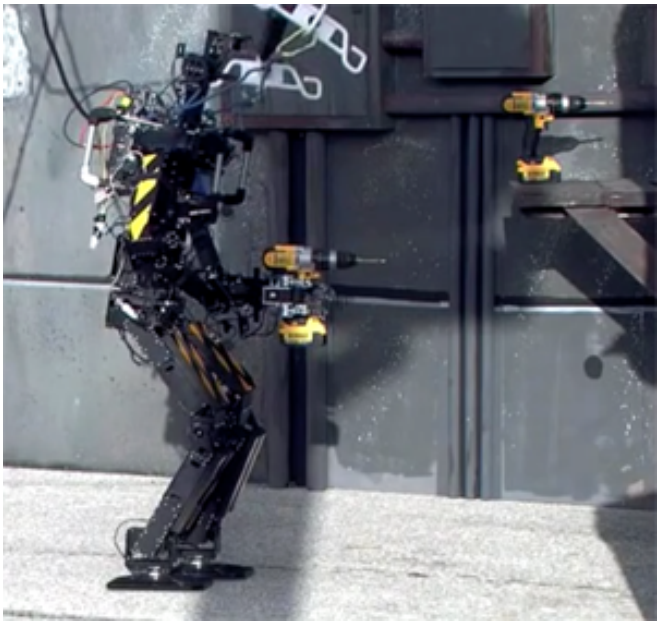


Fig. 1. The THOR-OP robot holds a drill in hand while walking stably.

With the limited time frame and varied operation requirements, we focused our development on modularity and cost effectiveness. Our hardware is very unique; all of the advanced actuators and structural parts are general purpose, standardized parts for the commercial market. The robot is assembled simply by bolting generic actuators and custom mounting brackets together, which allowed us to keep development cost very low and manufacture and repair speeds immensely fast. Finally, the estimated cost of our robot with only necessary sensors and hardware was much lower than we expected due to clever adaptable design.

Modularity in our software framework stemmed from the decision to extend our RoboCup humanoid soccer software framework [3] that controlled a number of small humanoid robots such as the Nao and DARwIn-OP. We were able to quickly port our software to the new THOR-OP (Tactical Hazardous Operations Robot, Open-Platform), including a wheeled intermediate THOR-OP version - THOR Centaur, by swapping only the hardware interface module, a configuration file, and the kinematics definitions. Similarly, we could simulate the robot using the Webots [4] robotics simulator with minimal changes of the code for other plat-

* GRASP Lab, University of Pennsylvania, Philadelphia, PA 19104
{yiseung,smcgill3,vlarry,heqin,ddlee}
@seas.upenn.edu
† Robotis, Co. Ltd. Seoul, Korea dudung@robotis.com
‡ Virginia Tech, Blacksburg, VA 24061 mrouleau@vt.edu
‡† University of California, Los Angeles, CA 90095
dennishong@ucla.edu

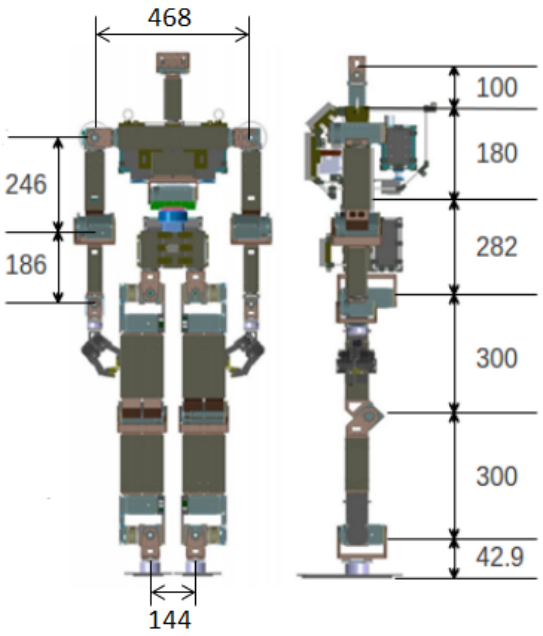


Fig. 2. Front and side views with dimensions (unit: *mm*) of a THOR-OP drawings with 6 degree of freedom arms.

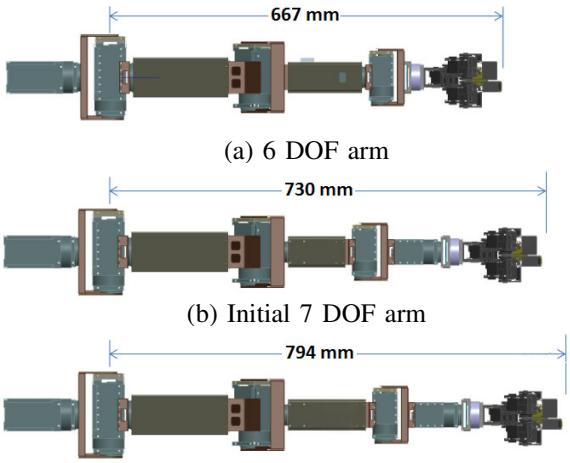
forms. This reusable and extensible nature drove our inter-process software stack, where multiple methods could access sensor feeds and robot commands. With a focus on state machines, we divided tasks into smaller projects to allow for parallel development without complete knowledge of the whole system. Finally, we developed multiple hierarchical modules for locomotion and manipulation control that can be dynamically selected to suit the current situation.

The remainder of the paper proceeds as follows. Section II describes the hardware platform we used for the DRC Trials. Section III explains the overall software structure and the interface. Section IV discusses the perception system including both visual and audio feed. Section V and VI describe our motion controllers, which includes the sampling based arm motion controller and the hybrid Zero Moment Point (ZMP) based locomotion controller with push recovery. Section VII describes our user interfaces to remotely operate the robot. Section VIII presents results at the DRC Trial held December 2013. Finally, we conclude with a discussion of potential future work.

II. Hardware Architecture

The THOR-OP robot used by Team THOR at the DRC Trials in December 2013 consisted of 31 actuators, 7 in each pair of arms, 6 in each pair of legs, 2 in the torso, 2 for the head, and 1 for panning the chest LIDAR. The robot stands 1.47m tall, weighs 49kg and has a wingspan of approximately 1.95m. The arms and legs have shock absorbing padding to protect the robot in the event of a fall and there is a roll cage around the upper body both for protecting the sensors and computer, and for ease of handling.

Fig. 3. The structural components for a single THOR-OP robot use standardized dimensions.



(a) 6 DOF arm

(b) Initial 7 DOF arm

(c) Final 7 DOF arm

Fig. 4. Three different arm configurations evolved during development for the THOR-OP. The 7th DOF was added to wrist for better dexterity.

A. Modular Actuator and Structural Components

For actuators, the new series of Dynamixel Pro servo-actuators developed by Robotis, Co. Ltd are used at most joints. THOR-OP uses three different motor types: H42-20-S300-R, H54-100-S500-R and H54-200-S500-R. They are rated at 20W, 100W and 200W 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 in-line output axle and one with parallel output axle. Both gearboxes use cycloidal reduction gears that have higher impact tolerance than common harmonic drives. Communication with the actuators is done via four RS485 buses.

In addition to the servo-actuators, the robot is mainly built with standardized structural components that are designed to be used with the Dynamixel Pro servo-actuators. They are extruded aluminum tubings and brackets with regularly spaced bolt holes, and one can easily put them together with hex bolts. The total man-hour needed for a complete assembly from parts is estimated at 24 hours. With the benefit of the modular construction of the robot, we could quickly iterate through a number of different designs. Figure 4 shows the the evolution of arm design over the course of our testing. The request of changes in DOF and link lengths based on

Fig. 5. The gripper for the DRC Trials consisted of two under actuated fingers to wrap around an object, and a rigid palm to give support.

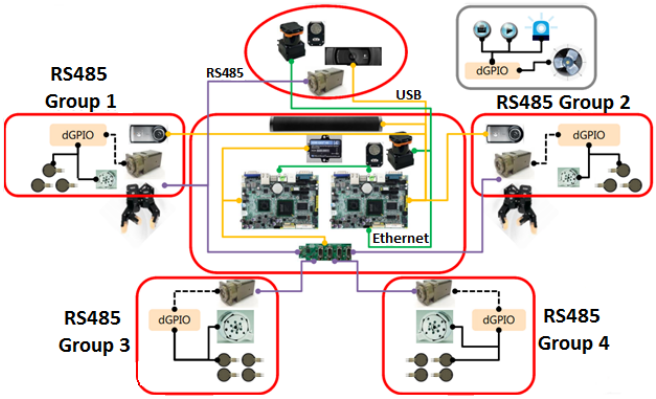


Fig. 6. The overall structure of the THOR-OP hardware interconnections includes four independent motor communication buses.

simulations and test experience were able to be completed in a very short time by our mechanical team.

B. End Effector

One of the few non-standard parts we used for the robot is the end effector. Over the course of our preparation for the DRC trials, we iterated over multiple gripper designs, and the final design incorporated two active fingers controlled by small Dynamixel MX-106R servomotors and a passive palm at the other side. Each finger has a passive joint with a spring-loaded linkage mechanism [5] to allow the finger to conform to a varied range of objects including drills, hoses and wooden blocks [6]. Each hand, like the one shown in Figure 5, weighs 767 grams.

In addition to the gripper hand, we used a number of passive end effectors for tasks that did not require finger actuation. One such end effector was a two-spoke mechanism that was used to align at the center of the valve and allow continuous rotation of the wrist for completing the valve task without having to reposition the body. Yet another simple mechanism was a long angled hook that was designed for opening the door without the need to grip the handle with the gripper.

C. Electronic Systems

The overall system structure and interfacing on THOR-OP is shown in figure 6. Two Axoimtek computers with 1.65GHz dual core AMD G-series APUs were available to perform high level computation, although only one was used in competition. Communication from these computers to the actuators was divided into four independent chains of RS485s over one USB to RS485 interface board. The robot operates on 24V for computers and actuators, saving for the LIDAR motor which requires a down-converted 12.5V supply. The maximum current the robot has drawn at any time is 20A; for this reason, the computer and the servo system reside on a separate supply to mitigate harmful current spike effects. With this separation, the emergency stop switch kills only the servos when required, leaving the computer unaffected.

D. 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. For more situational awareness, each wrist is equipped with a Logitech C905 Webcam. Also the robot is equipped with two ethernet based Hokuyo UTM-30LX-EW LIDAR sensors, one on the head and one in the chest. The head LIDAR, mounted horizontally, is mainly used to generate a 2D map using the SLAM algorithm described in [7]; however, this function was unused during competition. The chest LIDAR mounted vertically on a panning servo is used to generate a local 3D mesh of objects in front for manipulation. We have found that the LIDARs were not affected by direct sunlight even without any special shades or covers.

We used a Microstrain 3DM-GX3-45 inertial sensor located close to the center of mass of the robot, and utilized its raw accelerometer and gyro data, with extended Kalman filtered pose estimates. The robot also has one ATI Mini58 force-torque sensor and four Interlink FSR402 force-sensing resistors at each ankle, that are queried through external analog-in ports of the Dynamixel servos.

III. Software Architecture

There are many advantages to our modular software design. Each module performs logically discrete functionality, built separately from each other. When assembled together in a proper hierarchy, they constitute the application program. This type of system is very reusable and extendable; in fact, we reused many modules from our own open-source code base for RoboCup [3] to save time and effort.

We modularized our software architecture in several ways. First, we set aside processes to interface with each hardware device, including cameras, laser scanners, motors, and human interface devices. Each process had a twin sub-process in our simulation environment, such that minimal code modifications were needed to have the system run in simulation or on real hardware. Furthermore, we made a module that translated generic robot commands and queries into robot specific kinematics and motor packets, with the intention that a new robot platform needs only a similar module

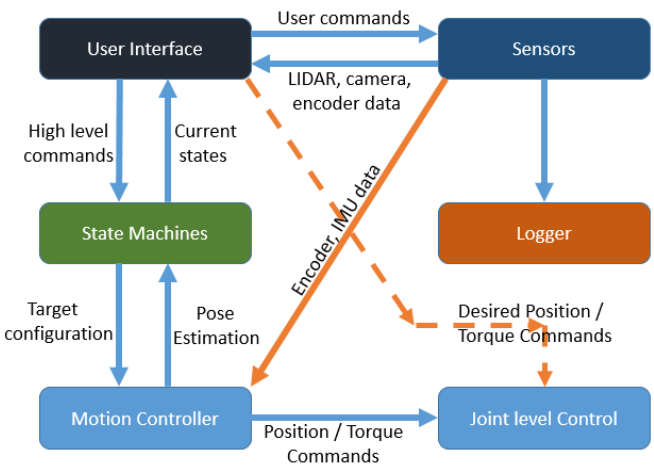


Fig. 7. The overall control system layout provides important separation of modules.

conforming to the API. Next, we constructed state machine modules that ran independently in a behavior process – ones for the overall body, the arms, the legs, the head, and the laser scanner. Finally, we formed operator modules to communicate data and commands with the robot, such that user interfaces could be mixed and matched.

A. Communication Architecture

In the DRC competition, the bandwidth and latency over the IP network alternated each minute between good and bad conditions. The good communication condition was 1Mbps bandwidth paired with 100 ms round trip delay, while bad communication was 10Kbps paired with a 1000ms. Due to variable bandwidth provided, we allowed for operator-specified on-the-fly configurable compression techniques and transmission frame rates of the camera and LIDAR data.

On board the robot, communication channels included Boost shared memory segments [8] and ZeroMQ message queues [9]. While these inter-process channel assets would reside on one single computer, we leveraged ZeroMQ's request/reply method (and UDP fallbacks) to provide a remote operator with access. For instance, in cases where the inverse kinematics solver was not able to find solutions for arm trajectories due to odd arm configuration or limits on range of motion being reached, the operator could manually override high-level arm configuration or joint angles. The user could dynamically select between UDP and ZeroMQ's TCP PUB/SUB methods. All metadata was efficiently serialized via MessagePack [10].

We tested extensively with settings on the MiniMaxwell network-shaper that ultimately was used in competition. We tested our network setup under more duress than the competition would provide, dropping 25% of packets, enabling reordered packets, and doubling the lag to 2000ms round trip times. We were always able to communicate effectively with the robot and observed no incorrect behavior. Our testing provided a larger degree of safety certainty and assurance of at least network robustness.

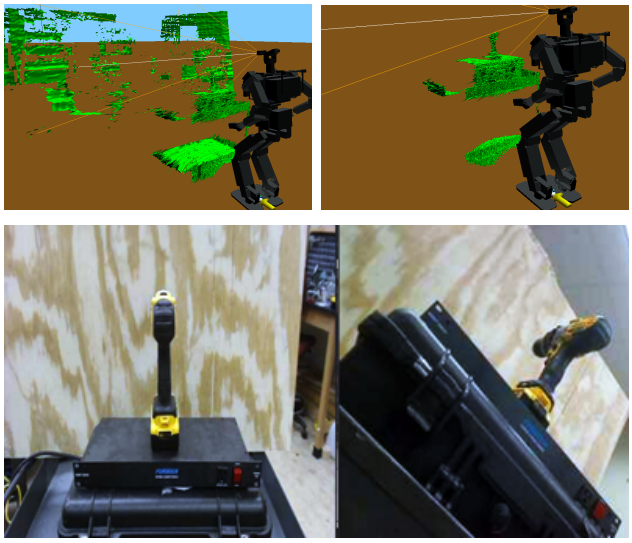


Fig. 8. Fast mesh (top left) and slow mesh (top right) methods provided 3D constructions from the LIDAR. Head (bottom left) and hand (bottom right) cameras provided two important perspectives for manipulation.

IV. Perception System

The perception system was 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 included a nine axis GPS-enabled IMU, multiple RGB cameras at head and wrists, two LIDARs, two force-torque (FT) sensors at that ankles and joint encoders in every joint. Among those sensors, the IMU, joint encoders and FT sensors are used constantly for state estimation and balancing of the robot; all others provide on-demand data for operator, depending on need and network quality.

A. 3D Reconstruction

We used only the Hokuyo LIDAR located within the chest of the robot. The sensor scanned vertically while panning horizontally. The depth data from this LIDAR was utilized to produce a 3D mesh of the environment and objects near the robot. The human operator was then able to locate the targets in 3D space and set up reference frames for motion planning. In practice, we used 90 degree and 60 degree vertical and horizontal fields of view, respectively.

To adapt to the limits on bandwidth usage, we used two different settings to filter and compress the LIDAR readings. For navigation we used "fast mesh" which was relatively noisy and showed surroundings within a range of 5 meters. This coarse image could be frequently requested without incurring network penalties. "Slow mesh" showing only close objects with much finer resolution was used during manipulation since it provided more accurate information.

B. 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

TABLE I

SENSOR STREAM SETTINGS

Visual	**Compression format**	**Quality** (0-100)	**Interval** (Hz)	**Frame Size** (kB)
Fast Mesh	JPEG	90	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

proved extremely helpful for turning the valve and gripping the drill, which constantly required precise depth perception.

The quality of both the head and hand cameras were able to be specified by user through shared memory variables. We did find that when higher resolutions were used with the poor network profile, packets would begin to overflow the network, and we incurred latencies of around 10 seconds until the good network profile was activated. Due to this, we kept our camera settings very conservative through the competition, as shown in Table I. In general, low resolution and low frequency provided adequate operator awareness of the environment and end effector positioning during the competition.

C. Audio Feedback

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

V. MANIPULATION CONTROL

The THOR-OP has a total of 34 degrees of freedom that are controlled by two separate controllers: the upper body motion controller for the arms, neck and waist and the lower body motion controller for the leg joints. The two controllers function separately, which enables the robot to locomote during manipulation, yet they are coupled so that the robot can keep balance in spite of different arm configurations. Here we describe the upper body motion controller that moves the neck, arm and waist joints for various manipulation tasks.

A. High Level Arm Control

High level arm control is done by specifying the reference trajectory of the end effector in Cartesian space, configured by a list of arm motion commands. The commands include the movement in Cartesian space as well as rotation, while fixing the wrist position and updating direct joint control. In addition to those basic arm commands we also define more complex parameterized trajectories for manipulation tasks such as opening the door. To prevent the arm from getting stuck at a bad configuration, the arm planner is designed to make sure that every planned arm trajectory is allowed to be executed only after confirming it is feasible.

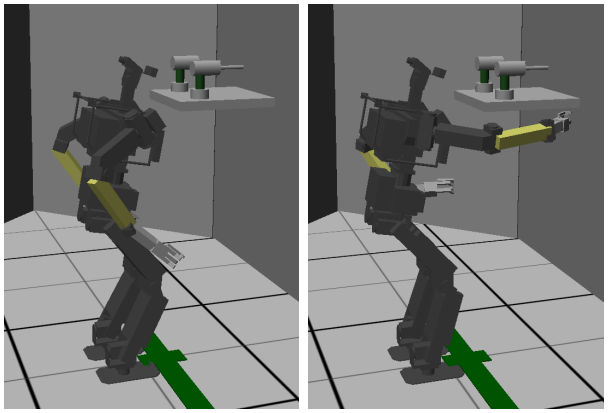


Fig. 9. The quasistatic full body balancing control moved the torso significantly to offset the weight of the arms.

B. Arm Planning

We use a simple sampling based approach to control the arms. We define the reference trajectory as a list of arm commands, and sample a number of points along the trajectory in 6D space. At each point, we sample a number of different shoulder yaw angles first, and used a closed form inverse kinematics solution for 6DOF arm to generate remaining six joint angles for each shoulder yaw angle. Finally, we find a locally optimal path that connects the sampled points to generate the joint-level keyframe for the arm movement. Then, the arm motion is executed by interpolating the points in joint space.

C. Full Body Balancing Controller

One tricky part for manipulation tasks is that different arm posture changes the mass distribution, and the robot must compensate for it to keep balance, usually by moving the torso or using push recovery techniques [11]. Yet, in order for the end effector to reach its target position, the arm postures also need be adjusted to compensate for the shifted torso position. Our approach maintains two torso COM positions: the virtual one, which coincides with the COM when arms are in the initial positions, and the actual one, which balances the robot. At every step, the upper body COM is calculated using the forward mass model of the robot, and the gripper target positions are compensated to take the difference between virtual and actual torso position. Finally, the difference is used in the lower body motion controller to offset the pelvis position to keep the COM of the entire body at the center of the support polygon.

VI. LOCOMOTION CONTROL

Overall, the locomotion of the robot, shown in Figure 10, can be controlled in three different modes. The *direct* mode, which uses the latency-free analytic ZMP controller, can control the velocity of the robot for every step. The next step position is calculated based on the current foot

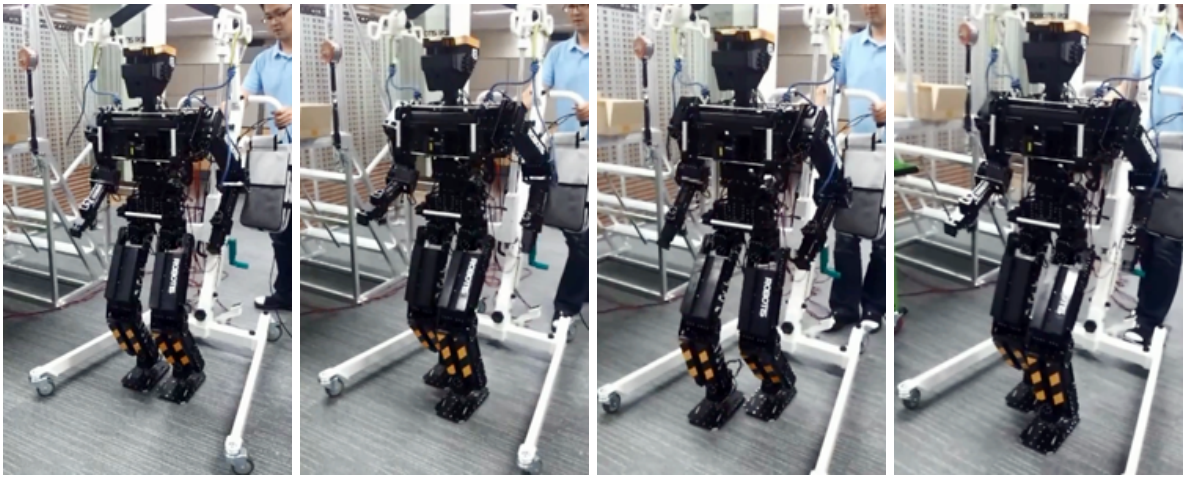


Fig. 10. THOR-OP robot demonstrating dynamic bipedal walking capability

configuration, the target velocity and kinematic constraints. The *preview* mode uses the target pose of the robot to generate a number of optimal steps to reach the pose, and uses the preview controller for locomotion. Finally, the *special* mode is involved for special cases such as stepping over a block. For each mode, the start and end of locomotion is handled by a preview controller for a smooth transition between standing and walking states.

A. Hybrid Walk Controller

We use a hybrid locomotion controller [12] which can 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 a less computationally expensive, latency-free locomotion pattern best suited for direct teleoperation, as well as a standard preview based walking using future foothold positions.

B. Push Recovery Control

To cope with unexpected perturbations, we have added two different types of push recovery control [13] that use the current state estimation from the IMU and joint encoders. An ankle strategy is implemented by controlling the target joint angle of the position controlled ankle joint, which generates a control torque as a result, and the stop reflex strategy stops walking and lowers the center of mass to resist large perturbation.

VII. Operator Console

We designed the remote operator interface in consideration of changing network situations, variable human assistance levels, and modular subsystems for online GUI modifications. We felt that it was incredibly important for a trained operator of the robot to be able to access any available resource on the robot at a moment's notice, and kept this design paradigm in mind. Ultimately, we utilized both a web page interface and a slew of terminal based programs to guide the robot through the variety of tasks.

A. Operator Console Setup

The basic operator setup is shown in Figure 12, which includes a laptop, an external display and an iPad. The monitor displays the head camera feed, while the iPad takes touch inputs for gripper control and displays the visual feedback of 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 the second laptop to monitor various processes of the robot and provide low level control of the robot via SSH sessions. In this configuration, we were able to utilize all levels of granularity for controlling our robot.

B. Command Line Interface

The basic mode of interacting with the robot was through an interactive Lua shell over SSH that provided raw access to shared memory and state machines events. In case a higher level controller fails, we can access a very low level behaviors of the robot in this shell. Another mode of controlling allows for task-specific commands via more refined terminal scripts, where we used specific key presses to directly initiate state machine events or manually override target positions of the end effectors, accessed over SSH. Different scripts were constructed for various purposes (locomotion, manipulation, or even a single task like driving) so that interference between sub processes was reduced to a minimal level during both test and competition.

C. Graphical User Interface

Command line scripts lack an ability to display rich sensor information; to this end, we designed an HTML5 based graphical user interface (GUI). Browser support allowed for cross platform development and during the competition; we operated iOS, OSX, and Linux platforms simultaneously. One master NodeJS [14] server bridges ZeroMQ messages and raw UDP packets from the robot to WebSockets messages for the browser. The browser's JavaScript console helped in identifying network conditions and in overall debugging. In fact, the JavaScript, CSS, or HTML code was occasionally modified on-the-fly during a run.

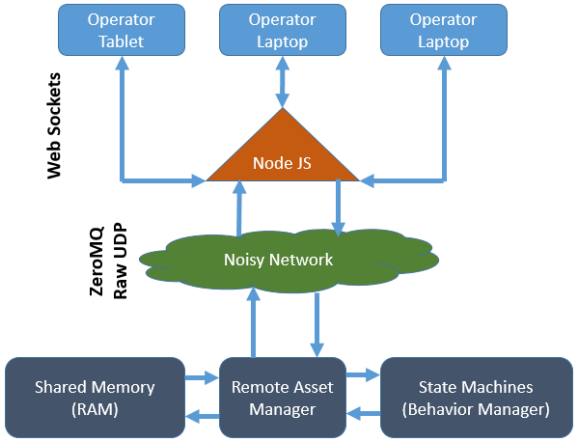


Fig. 11. The system layout for the operator interface allowed multiple machines to be used simultaneously.

Fig. 12. The operator interface setup included three main screens for the user to guide the robot and observe its environment.

A WebGL canvas using the THREE.js [15] framework delivered 3D views, shown in Figure 8, of both LIDAR returns and the robot's configuration given joint position feedback and inertial readings. This framework allows the user to set world coordinates for waypoints or end effector positions with good awareness. We triangulated LIDAR returns into a mesh similar to methods described in [16], with additional filters for the ground. While the computation cost was high for mesh fabrication, LIDAR readings were broadcast much slower than the computation time, and the operator perceived no lag in GUI experience since a WebWorker performed the calculation in a background thread.

VIII. Results

THOR-OP's modularity in both hardware and software was put to the ultimate test at the end of December, 2013 when the DRC Trials were held at the Homestead Motor Speedway in Florida. THOR-OP represented Team THOR and took front stage to perform quite well at the Homestead speedway. THOR-OP attempted all eight tasks set forth by the DRC and competed well, only succumbing to a couple unexpected hardware, software and setup logistical issues. Overall, we accrued 8 points and finished in 9th place out of 16 teams.

We had a few cases of damaged actuators during the trial – one just minutes before the ladder task – but thanks to the "modular" nature of the robot's hardware, we replaced the damaged actuator within minutes and went on to score on the task. This also allowed us easily use task-specific appendages, which greatly helped THOR-OP handle manipulation tasks more efficiently.

On the software side, we have found our locomotion controller with push recovery was quite capable in practice, which kept the robot standing upright in spite of various perturbations that made many other robots fall down and fail to complete the tasks. Additionally, our layered arm controller proved to be very competitive and robust in spite of its simplicity. Evidence of our stability and robustness was found in the valve task, where we were awarded "Best In Task" for fastest completion time among all teams.

Finally, we have found that our software framework demanded surprisingly little amount of computational power, averaging only 25% of the CPU time of a single on-board computer. A graphical comparison of the DRC teams and their performances at the trials can be seen in figure 14 (courtesy of the DRC organizing committee).

IX. Conclusions

We 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 heavily focused on modularity of both hardware and software structures. Important benefits included rapid field repairability of the robot, low development and manufacturing cost, and reduced development time – all vital aspects for any robotic approach to disaster response. The Trials results show that our hardware and software platform is very capable and we look forward to the finals next year for an even better performance from THOR-OP. 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 also acknowledge the support of the ONR SAFFIR program under contract N00014-11-1-0074.

References

[1] United States Department of Defense, "DARPA Robotics Challenge," Apr. 2012. [Online]. Available: http://go.usa.gov/mVj
[2] "Atlas - the agile anthropomorphic robot," 2013. [Online]. Available: http://www.bostondynamics.com/robot_Atlas.html
[3] S. G. McGill, J. Brindza, S.-J. Yi, and D. D. Lee, "Unified humanoid robotics software platform," in *The 5th Workshop on Humanoid Soccer Robots*, 2010.
[4] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004.
[5] T. Laliberte, L. Birglen, and C. Gosselin, "Underactuation in robotic grasping hands." *Machine Intelligence and Robotic Control*, vol. 4, no. 3, pp. 1–11, 2002.

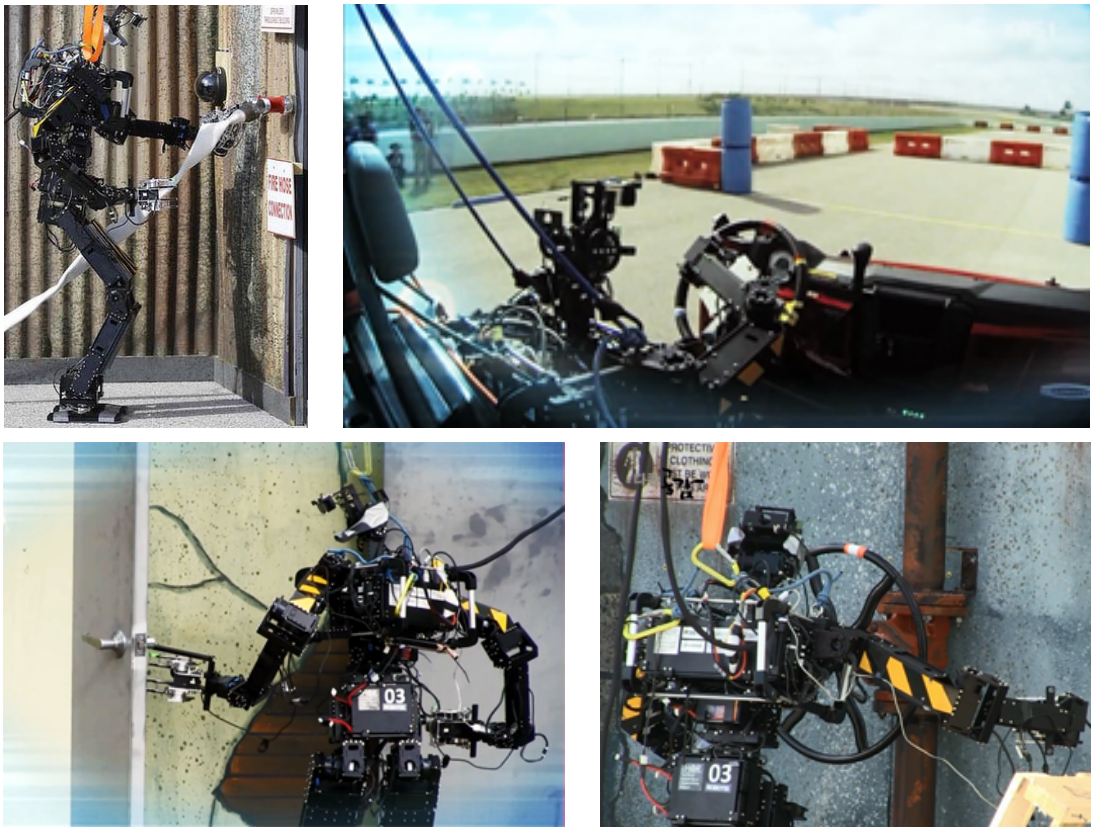


Fig. 13. THOR-OP at the 2013 DRC Trials performing four of five tasks in which it scored points.

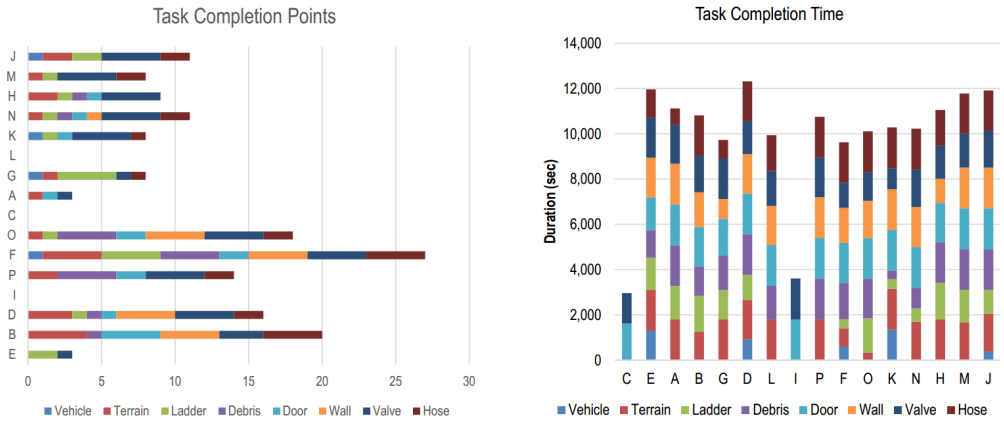


Fig. 14. Teams performances match-up, where 'K' denotes the Team THOR.

[6] M. Rouleau and D. Hong, "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*, 2014, submitted, under review.

[7] J. Butzke, K. Daniilidis, A. Kushleyev, D. D. Lee, M. Likhachev, C. Phillips, and M. Phillips, "The university of pennsylvania magic 2010 multi-robot unmanned vehicle system," *Journal of Field Robotics*, vol. 29, no. 5, pp. 745–761, 2012. [Online]. Available: http://dx.doi.org/10.1002/rob.21437

[8] Boost c++ libraries. [Online]. Available: http://www.boost.org

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

[10] Messagepack serialization library. [Online]. Available: http://www.msgpack.org

[11] S.-J. Yi, S. G. McGill, B.-T. Zhang, D. Hong, and D. D. Lee, "Active stabilization of a humanoid robot for real-time imitation of a human operator," in *IEEE-RAS International Conference on Humanoid Robots*, 2012, pp. 761–766.

[12] S.-J. Yi, D. Hong, and D. D. Lee, "A hybrid walk controller for resource-constrained humanoid robots," in *IEEE-RAS International Conference on Humanoid Robots*, 2013.

[13] S.-J. Yi, B.-T. Zhang, D. Hong, and D. D. Lee, "Online learning of a full body push recovery controller for omnidirectional walking," in *IEEE-RAS International Conference on Humanoid Robots*, 2011, pp. 1–6.

[14] Node.js webserver. [Online]. Available: http://www.nodejs.org

[15] R. Cabello. Three.js javascript webgl library. [Online]. Available: http://www.threejs.org

[16] D. Holz and S. Behnke, "Fast range image segmentation and smoothing using approximate surface reconstruction and region growing," in *Intelligent Autonomous Systems 12*. Springer, 2013, pp. 61–73.