Team IHMC's Lessons Learned from the DARPA Robotics Challenge Trials

Matthew Johnson, Brandon Shrewsbury, Sylvain Bertrand, Tingfan Wu, Daniel Duran, Marshall Floyd, Peter Abeles, Douglas Stephen, Nathan Mertins, Alex Lesman, John Carff, William Rifenburgh, Pushyami Kaveti, Wessel Straatman, Jesper Smith, Maarten Griffioen, Brooke Layton, Tomas de Boer, Twan Koolen, Peter Neuhaus, and Jerry Pratt
IHMC Robotics, 40 South Alcaniz Street, Pensacola, Florida 32502
e-mail: mjohnson@ihmc.us, bshrewsbury@ihmc.us, sbertrand@ihmc.us, twu@ihmc.us, dduran@ihmc.us, mfloyd@ihmc.us, pabeles@ihmc.us, dstephen@ihmc.us, nmertins@ihmc.us, alesman@ihmc.us, jcarff@ihmc.us, wRifenburgh@ihmc.us, pKaveti@ihmc.us, wStraatman@ihmc.us, jsmith@ihmc.us, mGriffioen@ihmc.us, blayton@ihmc.us, tdeboer@ihmc.us, tkoolen@ihmc.us, pneuhaus@ihmc.us, pratt@ihmc.us

Received 14 March 2014; accepted 23 November 2014

This article is a summary of the experiences of the Florida Institute for Human & Machine Cognition (IHMC) team during the DARPA Robotics Challenge (DRC) Trials. The primary goal of the DRC is to develop robots capable of assisting humans in responding to natural and manmade disasters. The robots are expected to use standard tools and equipment to accomplish the mission. The DRC Trials consisted of eight different challenges that tested robot mobility, manipulation, and control under degraded communications and time constraints. Team IHMC competed using the Atlas humanoid robot made by Boston Dynamics. We competed against 16 international teams and placed second in the competition. This article discusses the challenges we faced in transitioning from simulation to hardware. It also discusses the lessons learned both during the competition and in the months of preparation leading up to it. The lessons address the value of reliable hardware and solid software practices. They also cover effective approaches to bipedal walking and designing for human-robot teamwork. Lastly, the lessons present a philosophical discussion about choices related to designing robotic systems. © 2015 Wiley Periodicals, Inc.

1. INTRODUCTION

The Florida Institute for Human & Machine Cognition (IHMC) in Pensacola, Florida recently competed in the DARPA Robotics Challenge (DRC). The DRC is an international robotics competition hosted by the Defense Advanced Research Projects Agency (DARPA). The primary goal of the DRC is to develop robots capable of assisting humans in responding to natural and manmade disasters. The robots are expected to use standard tools and equipment used by humans to accomplish the mission.

The first event of the DRC was called the Virtual Robotics Challenge (VRC), which was held in June 2013. The top performers[1] in this phase were each provided with an Atlas humanoid robot, made by Boston Dynamics, and funding to continue research and development for the next event. The second event phase of the DRC was the DRC Trials, which was held in December 2013 at the Homestead-Miami Speedway in Florida. There were 16 teams from around the world competing at the Trials: the top eight teams of the VRC used the provided Atlas robot, and the rest competed using a robotics platform that they purchased or built on their own.

IHMC placed 1st in the VRC[2] and 2nd in the DRC Trials.[3] Much of our success is due to our years of research on bipedal locomotion, our approach to effective human-machine interaction, and robust software practices. This article discusses the challenges we faced in transitioning from simulation to fielded hardware. It also discusses the lessons learned both during the competition and in the months of preparation leading up to it.

[1]http://www.darpa.mil/NewsEvents/Releases/2013/06/27.aspx (accessed 18AUG2014).

[2]http://www.darpa.mil/NewsEvents/Releases/2013/06/27.aspx (accessed 18AUG2014).
[3]http://www.darpa.mil/NewsEvents/Releases/2013/12/26.aspx (accessed 18AUG2014).

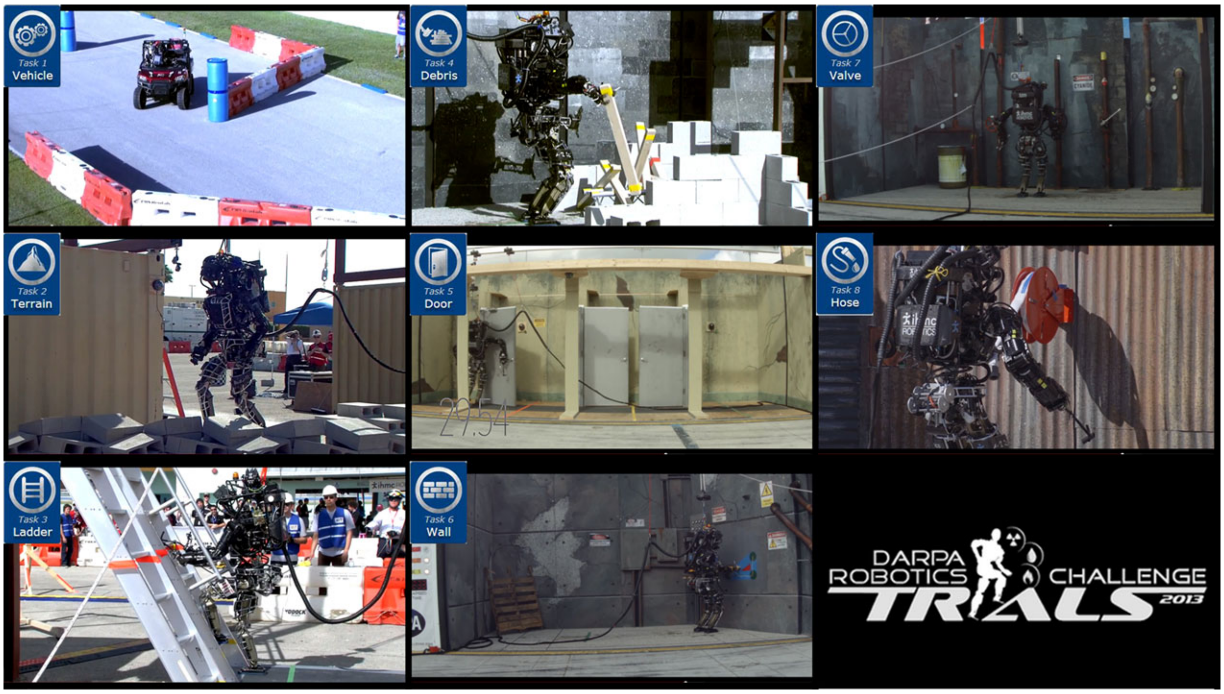

Figure 1. Images of the actual DRC Trials tasks (all images are pulled from DRC videos on the DARPA YouTube channel[4]).

2. BACKGROUND

2.1. DRC Trials

The DRC Trials consisted of eight tasks (Figure 1) designed to mimic actions performed by first responders in disaster environments. The tasks included the following:

1. Vehicle task: drive a Polaris Ranger through an obstacle course and egress to an exit.
2. Debris task: remove 10 pieces of debris from a doorway and then enter the doorway.
3. Valve task: close three valves (3 in. ball, 9 in. gate, and 18 in. gate).
4. Terrain task: walk over ramps, cinder blocks, steps, and uneven angled cinderblocks.
5. Door task: pass through three doors (push, pull, and weighted self-closing pull).
6. Hose task: pick up a fire hose and attach the connector to a spigot.
7. Ladder task: ascend ladder approximately 2.4 m high.
8. Wall task: use a cordless tool to cut a shape out of sheetrock.

For each task, a robot is allotted 30 min to complete the objective. The robot operator was required to be out of the line of sight of the robot at all times. Additionally, DARPA provided bandwidth constraints by introducing a network shaper that oscillated throughput and latency between "good" (1 Mb/s and 100 ms delay) and "bad" (100 kb/s and 1,000 ms delay) communications. Each task had an individual criterion for awarding a total of four points, with incremental scoring for the first three points and a bonus point for task completion without a fall or other intervention.

2.2. The Atlas Humanoid Robot

To complete the tasks, we used the Atlas humanoid robot from Boston Dynamics (Figure 2, left). Atlas is a hydraulically powered humanoid robot that weighs 150 Kg and stands 1.88 m tall. The robot has 28 degrees of freedom with six in each major appendage, one in the neck, and an additional three in the pelvis. Atlas requires 480 V three-phase power provided offboard by an extendable tether. A Carnegie Robotics MultiSense-SL head (Figure 2, right) provides two forward-facing cameras and an axial rotating Hokuyo LIDAR. The Atlas also has two wide-angle cameras intended to compensate for the robot not being able to yaw its head. An interchangeable end-effector mount terminates each arm, allowing third party hand installations. Three end effectors were provided with Atlas, and they are discussed in Section 3.3.2.

[4]DARPA TV YouTube channel https://www.youtube.com/user/DARPAtv/videos (accessed 18AUG2014).

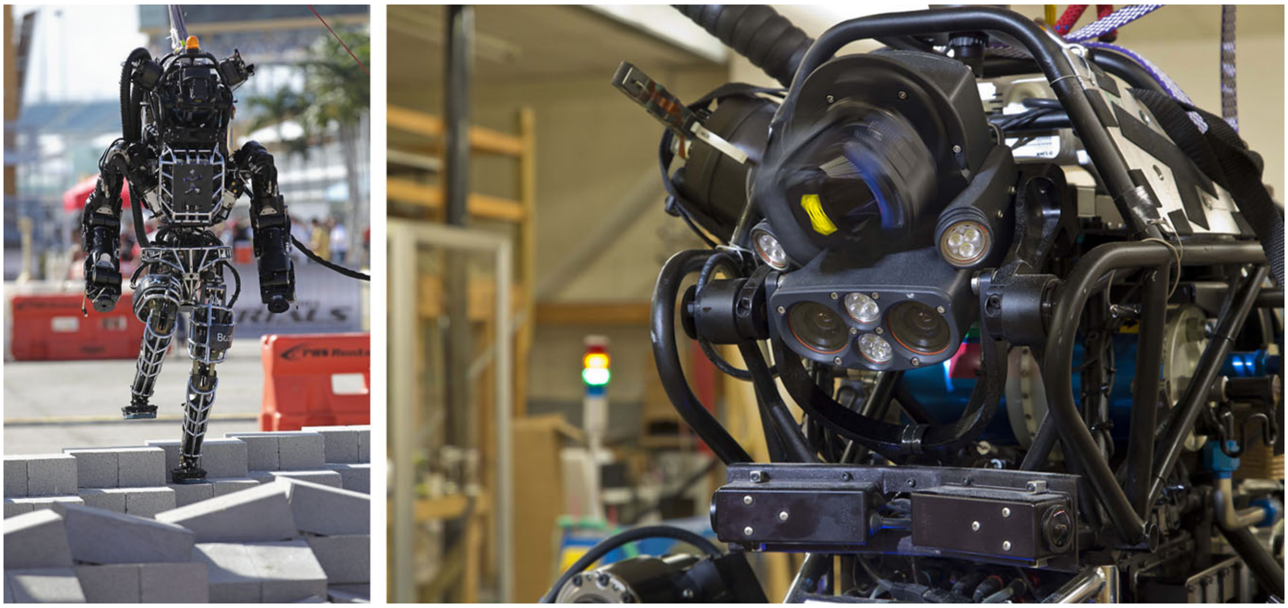

Figure 2. Atlas robot (left) provided by Boston Dynamics and a closeup of the Carnegie Robotics MultiSense-SL head (right).

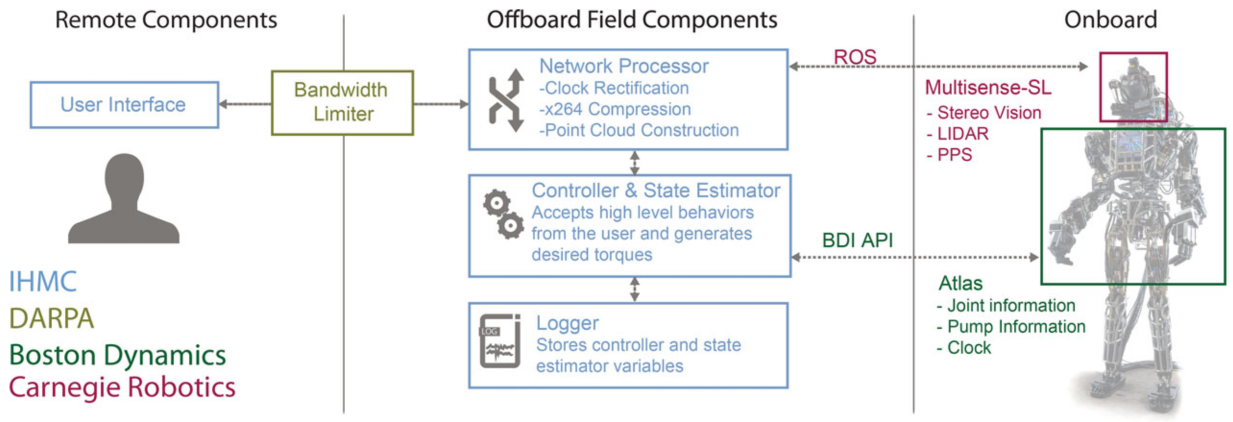

Figure 3. IHMC's System Architecture for the DRC Trials. Blue indicates IHMC code. Other colors indicate code and hardware provided by DARPA, Boston Dynamics, and Carnegie Robotics.

2.3. Team IHMC's System Architecture

A brief overview of the system architecture, shown in Figure 3, is provided for context. Boston Dynamics provided a low-level application programming interface (API) that allows control of joint positions and torques for each of the joints on the Atlas. They also provided an API for higher-level walking commands, but our team did not use this, preferring to use our own walking algorithm. The Carnegie head provided camera and LIDAR information via a robot operating system (ROS) interface. All teams using the Atlas were required to run their code offboard on what is called the "field" computer. This computer represents what would normally be running onboard. Onboard the field computer, IHMC ran its control algorithms and state estimation, network processing, and logging. The network processor was designed to manage our data flow in order to comply with DARPA-imposed bandwidth limits. It manages all data going to the operator. DARPA provided a bandwidth limiter to constrain communications to what might be available in a realistic operational scenario. The "operator" computer is a computer that was remote and out of the line of sight of the robot at all times. It is the computer that provided a user interface to the operator for controlling the Atlas robot.

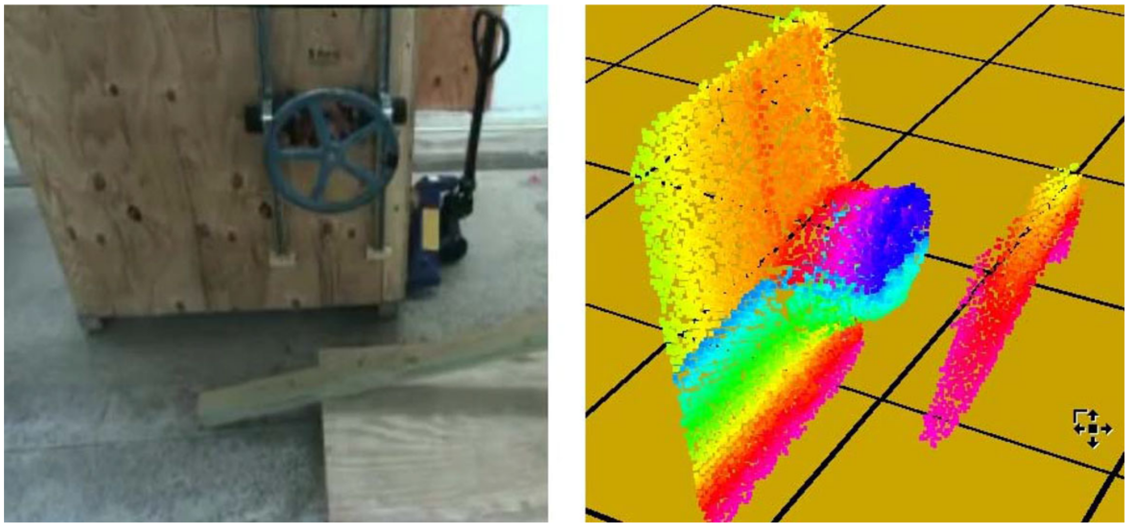

Figure 4. Left: Camera image of valve. Right: Point cloud showing LIDAR shadow. Note how edges of the valve have false points going back to the wall. The rainbow color depth map provides a mechanism for the operator to estimate the actual thickness of an object.

3. CHALLENGES IN MOVING FROM SIMULATION TO HARDWARE

The primary challenge in transitioning from the VRC to the DRC Trials was in understanding and addressing the differences between simulation and hardware. Several of the issues discussed are common issues in robotics, and others are specific to the Atlas or the competition. We will discuss how each issue impacted performance and what we did to mitigate the effects.

3.1. Sensing

For the DRC, the operator was required to be remote and out of the line of sight of the robot at all times. This meant that the operator's awareness was completely dependent on accurate and timely sensor data. This was not a problem for the simulated environment of the VRC, but it posed several challenges on the real Atlas hardware for the DRC Trials. These issues mainly dealt with sensor synchronization, sensor alignment, and sensor noise.

3.1.1. Sensor Synchronization

During the VRC, the LIDAR and camera output is perfectly synchronized with Atlas' data about robot state. The physical LIDAR and camera sensors in the MultiSense head operate independent of Atlas, requiring external synchronization to correlate with the robot state. These sensors are integrated within the MultiSense head, which provides a pulse per second (PPS) signal to both Atlas and the user to aid in synchronization. Once the PPS signal is received, we replace the PPS pulse time with the ROS clock time. This allows for the compensation of any clock drift between Atlas, the MultiSense head, and the ROS with each PPS tick. LIDAR sensor information is then correlated with the robot's sensor data, allowing for accurate placement of LIDAR points with respect to the robot as it moves.

3.1.2. LIDAR Shadows

While LIDAR data were perfectly clean in simulation, the actual Hokuyo LIDAR was susceptible to noise along the edges of objects. If the LIDAR beam hits multiple objects, then it returns a range between both objects. This structured noise appears as a shadow, as shown in Figure 4, and it makes the edges of objects difficult for the operator to see. This is especially true in cluttered environments, such as the debris removal task. As a result, accurately placing an end effector behind objects like a valve (e.g., Figure 4) was challenging. Our rainbow depth color scheme, shown on the right side of Figure 4, provided the operator with some ability to estimate thickness within the noisy data.

3.1.3. Joint Angle Offsets and Calibration

While joint angle offsets were not an issue in simulation, for Atlas there was an offset between the actual joint angles and the measured joint angles of up to 5°. The offset made fine manipulation tasks difficult.

To reduce the offset in joint angle measurements, we used common techniques for calibration involving a camera (Pradeep, Konolige, & Berger, 2014). In our case, we aimed to minimize the end-effector error measured by kinematic loop and a camera checkerboard pair as shown in Figure 5.

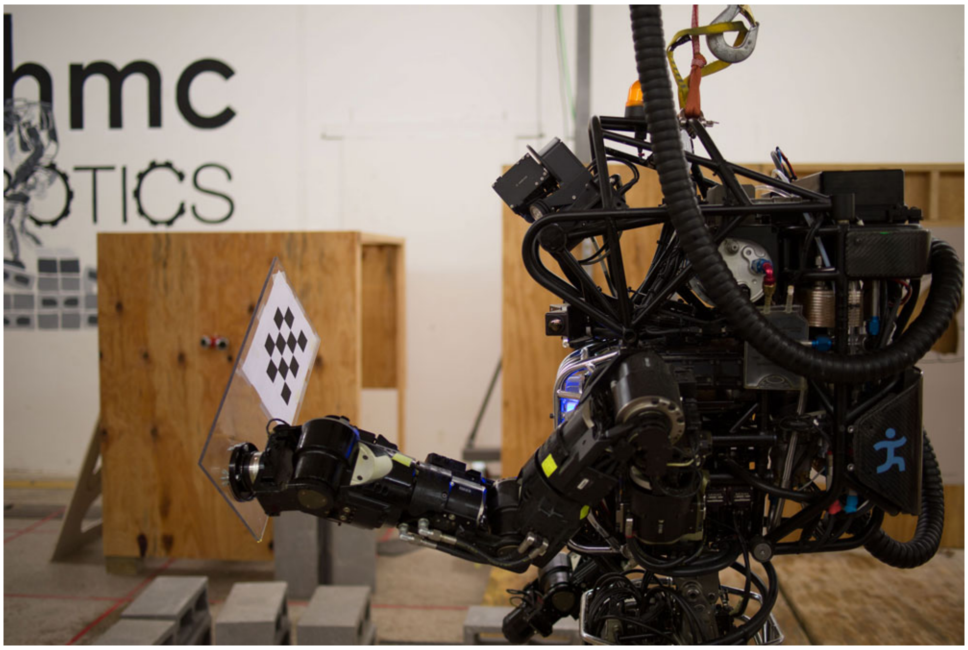

Figure 5. Joint calibration setup with checkerboard attached to robot arm.

Overall, the calibration reduced the joint angle offsets from approximately 5° to approximately 0.5° assuming the camera to be ground truth.

3.2. Control

The complexity of a humanoid robot makes it a daunting controls challenge. Our successful VRC control architecture required several extensions and modifications to address the real-world complexity of the Trials. An overview of our control architecture is shown in Figure 6. The gray areas in the figure indicate the areas that changed as we moved to physical hardware. Each will be discussed in detail in the following sections.

3.2.1. System Identification

In both the VRC and DRC, we were given the dynamic and kinematic models for Atlas. The VRC models were exact because the simulator uses an identical model for simulation; however, the model of the physical robot is only a best guess of the actual system. The Atlas is a proprietary system, and we were not provided access to everything we would like for complete system identification. We did partial system identification utilizing the redundancy of existing onboard sensors of the following two key parameters: torso center of mass and neck/arm joint angle offsets.

The torso represents approximately one-third of the total mass of the robot. Therefore, it plays an important role in predicting the center of mass (CoM) of the entire robot, which is crucial to our model-based walking and balancing algorithms. Dynamics identification was carried out using a linear least-squares method, matching the model-predicted center of pressure (CoP) and measured CoP using the foot-mounted force plate sensors. With data collected from 200 random balancing postures, the calibration reduced the average CoP error from 4 cm down to 1 cm.

3.2.2. Joint Sensor Processing to Account for Noise and Errors

The physical Atlas has joint sensor noise, flexible links, and joint backlash, whereas the virtual robot had none of these. The sensor processing module was created and added to the control architecture, as shown in Figure 6.

When implementing the control architecture on Atlas, the first step was to deal with the joint sensor noise. We have added low-pass filters that were tuned online on the real Atlas. Tuning filter parameters was a long process and required several iterations.

3.2.2.1. Link Elasticity

On the physical Atlas, link elasticity is non-negligible and can introduce an error of several centimeters on the location, for instance of the support foot or the pelvis. This was a critical issue and was a source of robot failures. The leg elasticity was identified in double support, i.e., both feet on the ground and not moving, using the correlation of the perceived foot displacement and the joint torques. To reduce

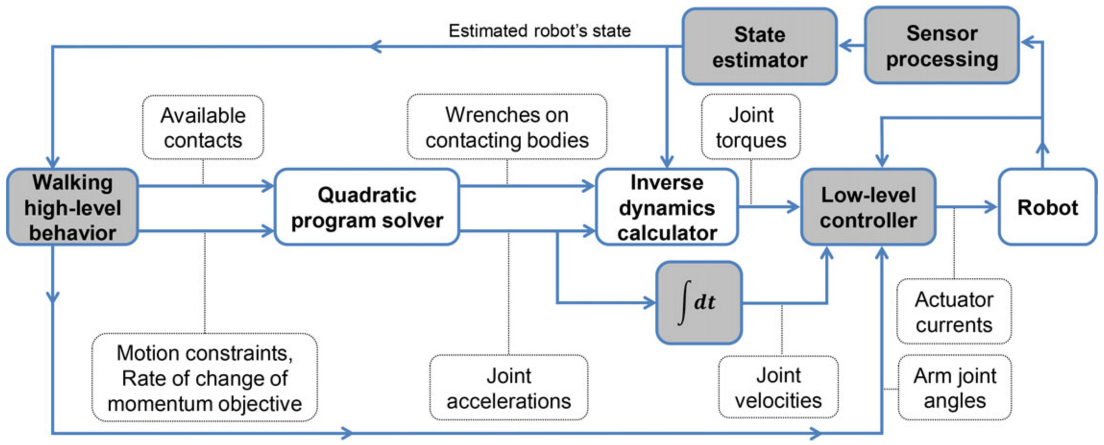

Figure 6. Overview of the information flow in the control architecture. The gray blocks refer to the modules that were added or modified when switching to the physical robot.

erroneous foot displacement, a compensator was added in the sensor processing for every leg joint as follows:

$$q_{\text{proc}} = q_{\text{raw}} + \tau/k,$$

where q_{proc} is the processed joint angle, q_{raw} is the raw joint angle, τ is the desired joint torque, and k is the stiffness coefficient that is tuned for each leg joint.

3.2.2.2. Joint Backlash

The joint position sensors, used for the feedback controllers, are located at the actuator output and do not sense the joint backlash. When the actuator moves within the slack, the sensed velocity measured can reach high values even when the link is barely moving. This perceived velocity causes instabilities in the controller that tries to dampen the perceived motion. This is also an issue when estimating the CoM velocity used in our balance controller.

A backlash compensator was added to the joint sensor processing. In addition to filtering the measurements, every time the joint motion changes direction, the compensator goes into a slop state. This state ends when the sign of the joint velocity is the same for a given duration; here we chose $\Delta t = 0.03$ s. During that time, the perceived velocity is multiplied by η. The scalar η starts out at 0 and linearly grows to 1 over the course of the slop state. The backlash compensator is expressed as follows:

$$\dot{q}^{t}_{\text{proc}} = \alpha \dot{q}^{t-1}_{\text{proc}} + (1-\alpha)\dot{q}_{\text{raw}}\eta$$

with:

$$\eta = \begin{cases} \dfrac{t - t_0}{\Delta t} & \text{if } (t - t_0) < \Delta t, \\ 1 & \text{otherwise,} \end{cases}$$

where t_0 corresponds the last instant t at which $\dot{q}_{\text{raw}}$ changed sign, $\dot{q}_{\text{raw}}$ is the raw joint velocity, $\dot{q}^{t}_{\text{proc}}$ and $\dot{q}^{t-1}_{\text{proc}}$ are the processed joint velocity at the current and previous step time, respectively, and α is computed to provide a break frequency of 16 Hz for the filter.

3.2.3. State Estimator

The modular state estimator (MSE) developed and used for the VRC (Koolen et al., 2013) provided a highly accurate estimation of the virtual robot. However, for reasons still unknown to us, it caused feedback instabilities in the control when running on the real robot. This issue was confirmed by swapping the MSE with a simple kinematics-based state estimator (KSE), which stabilized the controller. Rather than putting effort in fixing the MSE, we developed a new state estimator starting from the KSE. At the beginning, the KSE relied purely on the kinematics of the legs to estimate the pelvis global state. Step by step, we implemented new features that were individually tested and validated on both the virtual and real robots.

The KSE, in its current state, uses the inertial measurement unit (IMU) measurement to compute the pelvis orientation and linear velocity. It fuses the IMU and kinematics measurements to estimate the pelvis position and linear velocity as follows:

$$\dot{x}^{t} = \alpha_{\dot{x}}(\dot{x}^{t-1} + \ddot{x}^{t}_{\text{IMU}}\Delta t) + (1-\alpha_{\dot{x}})\dot{x}_{\text{kin}}, \tag{1}$$

$$x^{t} = \alpha_{x}(x^{t-1} + \dot{x}^{t}\Delta t) + (1-\alpha_{x})x_{\text{kin}}, \tag{2}$$

where x, $\dot{x}$, and $\ddot{x}$ are the pelvis position, linear velocity, and linear acceleration, respectively, superscripts t and $t-1$ refer to the current and previous step time, and the subscripts "IMU" and "kin" indicate whether the measurement is obtained from the IMU or the leg kinematics. We found that the weighting parameters $\alpha_{\dot{x}} = 0.992$ and $\alpha_{x} - 0.8$ resulted in the best state estimator performance, with the state estimator update time of 3 ms.

At high movement frequencies, the estimation of pelvis position and linear velocity is mainly based on the IMU acceleration output signal, limiting the influence of backlash contained in the kinematics measurement. At low frequencies, the usage of leg kinematics reduces the problem of drift coming from the integration of the IMU acceleration into linear velocity and position.

As opposed to the MSE used for the VRC, the KSE is independent of the system model and the high-level controller. For the MSE, the controller had to determine nonslipping contact points to be used for the estimation, whereas the KSE uses sensors, such as foot force sensors, to determine whether or not a foot is appropriate for the estimation. To limit errors resulting from unexpected foot rotation, the CoP is considered as the only nonslipping point in the foot and is used to estimate the pelvis state.

3.2.4. Walking High-level Behavior

Transitioning from the VRC to the DRC Trials, the main issues encountered in the walking behavior were knee collapse and chatter.

Atlas is susceptible to knee collapse when crouching, which results in a fall. We worked on singularity handling to reach a straight knee walking gait, and we put effort into our balance controller to robustly handle toe-off. This was sufficient to walk on flat ground or step on obstacles with minimal instances of knee collapsing.

Real-world imperfections, such as joint backlash and communication latency, would often lead to feedback instabilities in the controller. These instabilities can be observed through self-sustained high-frequency oscillations of the system, which we refer to as chatter. During the chatter, the links appear to reach physically unfeasible accelerations as perceived by the controller, whereas the actual robot is not moving at all. The main issue is that the chatter may amplify, leading to actual joint oscillations, finally resulting in physical instabilities and falls.

We addressed chatter by first reducing proportional derivative (PD) controller gains, especially damping gains given that they represent the main source of chatter and because the robot has strong passive damping. Lowering gains greatly reduced the frequency of chatter. We also limited the motion constraints to a maximum acceleration and a maximum jerk. For instance, we use a PD controller for the pelvis orientation. The range of acceleration and jerk necessary to achieve the walking motion was evaluated to properly limit the PD controller output, reducing the amplitude of the chatter. Modifying the function to minimize in the quadratic program solver also reduced chatter, as discussed in the next section.

3.2.5. Low-level Torque Controller

While the whole-body control framework computes the desired joint torques, the real robot can only be controlled through the current of the actuator valves. A low-level controller is implemented as an interface between high-level control framework and the robot. It consists of a closed-loop controller on joint torques resulting in the actuator valve currents to apply.

A closed-loop controller on torque was used to compute the desired valve currents for the real robot hydraulic actuators. On Atlas, the cylinder pressure measurements are used to evaluate the joint torque. Consequently, the stiction[5] coming from the sealing and the mechanical transmission between the actuator and the joint are hidden from the provided sensor data. In the worst case, we evaluated the stiction to be about 10 Nm. As a result, even with perfect force tracking, error in the control may emerge. This is due to stiction causing inaccuracy in the control framework.

To compensate for stiction, we chose to increase the control impedance by adding the desired joint velocity and position to the low-level control. The hybrid, i.e., torque-velocity-position controller, low-level control law can be written as follows:

$$i = k_{ff_{qd}}\dot{q} + i_\tau + i_{\dot{q}} + i_q$$

$$\text{with} \begin{cases} i_\tau = k_{\tau,p}(\tau_d - \tau) + k_{\tau,d}(\dot{\tau}_d - \dot{\tau}) + k_{\tau,i}\int(\tau_d - \tau)dt, \\ i_{\dot{q}} = k_{\dot{q}}\left(\int \ddot{q}_d dt - \dot{q}\right), \\ i_q = k_{q,p}(q_d - q) + k_{q,v}(\dot{q}_d - \dot{q}) + k_{q,i}\int(q_d - q)dt, \end{cases}$$

where $k_{ff_{qd}}\dot{q}$ is a feedforward term to compensate for the oil flow in the cylinder as the actuator is moving, i_τ is the desired current resulting from a proportional integral derivative (PID) controller on joint torque, $i_{\dot{q}}$ is the desired current from a P controller on joint velocity, and i_q is the desired current from a PID controller on the joint position.

For every joint but the arm joints, we use a hybrid controller on joint torques and velocities. The control law can be written as follows:

$$i = k_{ff_{qd}}\dot{q} + i_\tau + i_{\dot{q}}.$$

The desired joint velocities are computed by integrating from the desired joint acceleration (Figure 6). This hybrid controller greatly improved the overall motion tracking, but it also made the robot less compliant. During support, we switch to a pure low-level torque controller to increase the compliance of the ankle joints.

Having a higher stiction in the arm joints, we had to use a more rigid controller. The hybrid controller on joint torques and positions used can be written as follows:

$$i = k_{ff_{qd}}\dot{q} + i_\tau + i_q.$$

[5]Stiction is the static friction that needs to be overcome to enable relative motion of stationary objects in contact.

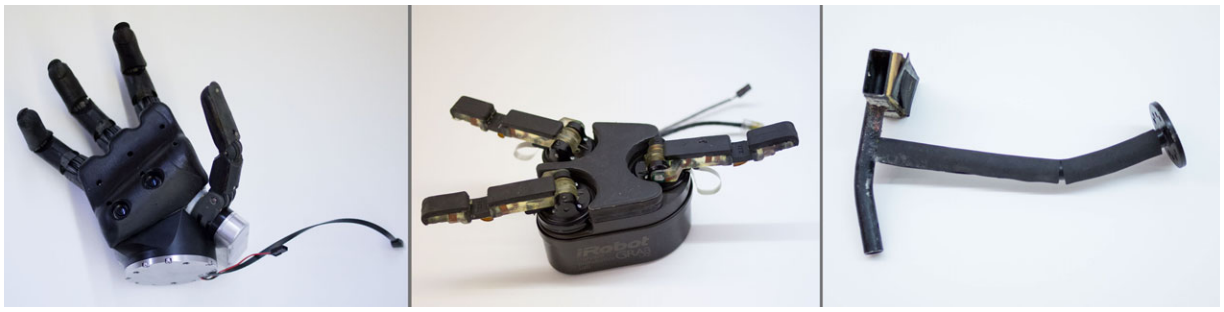
Figure 7. End effectors allowed for the DRC Trials. From left to right: Sandia hand, iRobot hand named Dexter, and an IHMC designed hook.

The desired arm joint angles q_d and velocities $\dot{q}_d$ are directly provided by the walking behavior. This low-level controller provided the best tradeoff between motion tracking and stability.

3.2.6. Real-time Control Issues

During the VRC, the simulation provided near real-time loops, running between 80% and 90% real-time rate. The simulation provided telemetry data to our controller at 1,000 Hz in simulation time. The competition rules were set up such that the controller had to send control data at a nominal rate of 200 Hz. Occasional delays of up to 20 ms were allowed to compensate for unexpected delays in the communication. In contrast, Atlas sends telemetry data to the robot controller at a constant rate of 333 ⅓ Hz. These data are handled during the next control tick. For our control architecture, where we directly command desired joint torques, effective performance required data updates every 6 ms.

Our controller is set up to have a dedicated state estimation thread and a control thread. The state estimation thread runs at the rate new data are coming in, 1,000 Hz for the VRC and 333 ⅓ Hz for the DRC trials. Our control thread runs at a slower rate to allow more computation time, 200 Hz in the VRC and 166 ⅔ Hz in the DRC Trials. During the VRC, we were able to use a non-real-time OS with the standard OpenJDK Java VM. By limiting unneeded object generation, we were able to reduce the garbage collection delays to be below 20 ms. However, for the real hardware we needed to switch to a system that guarantees that control signals are generated within the 6 ms deadline. We chose to base our system on Linux + RT PREEMPT patches. As none of the commercially available implementations of the Java Real Time Specification had the performance required to run our controller, we implemented a simple Posix Realtime Thread on top of the OpenJDK using JNI. This forced us to reduce object generation to a minimum, as a garbage collection cycle would exceed the deadline. In combination with an 18 GB heap size, we were able to run for 30 min. Furthermore, the default locking strategies in the OpenJDK do not protect against priority inversion,[6] so we avoided the use of locking and wrote our own non-blocking communication layers based on memory barriers, as described for the Disruptor.[7] A side effect of reducing object generation and removing all locks in our code was a significant reduction in execution time, reducing jitter and providing a buffer between execution deadlines and actual execution time.

3.3. Manipulation

The manipulation challenges in the DRC Trials were much more difficult than those of the VRC. They required a broader range of motion that pushed the limits of what is feasible with the Atlas robot. They also required finer grained manipulation, which proved challenging for the end-effector options available for the Atlas.

3.3.1. Range of Motion Limitations

While not directly related to transitioning from simulation to hardware, workspace limitations were accentuated by the additional tasks and the diversity of manipulation requirements the DRC Trials imposed. The Atlas arms offer a limited manipulation workspace, which was evident on both the virtual and physical Atlas. Tasks in the Trials that required manipulation across distances on the order of 0.6 m, such as the wall task, would require supplementing the arm motion with torso yaw. Similarly, tasks that had a vertical range of motion often required pelvis height adjustments. The workspace limitations were most evident for the debris clearing task. For the debris task, we could not reach low enough to the ground to grasp some of the debris required for the task. To compensate for the poor reach, we added 8 in. extensions between the hands and arms.

[6]See www.rtsj.org/docs/rtsj_1.0.2_spec.pdf (accessed on 09MAR2014).
[7]See http://lmax-exchange.github.io/disruptor/ (accessed on 09MAR2014).

3.3.2. End Effectors

Figure 7 shows the options for the arm end effectors provided by DARPA: (1) a four-fingered hand provided by Sandia National Laboratory,[8] (2) a three-fingered hand named Dexter based on the iHY hand provided by iRobot (Odhner et al., 2013), and (3) a metal hook. Each hand was evaluated according to its reliability, durability, and ease of use for the operator.

Starting with the Sandia hands, we identified a series of issues that prevented us from reliably accomplishing the manipulation tasks. The Sandia hand is bulky, making it hard to see what it is grasping, and the shape of the palm combined with the location of the fingers results in a very unreliable grasp with poor grasp strength for the tasks we were attempting in this competition.

The iRobot Dexter hand provided a strong grasp with reasonable visibility of the object being manipulated. The main issue identified with this hand was the durability. The fingers are driven using tendons that were prone to break after repeated use. An additional issue was reliability. The hand could exceed the 5 amp current limit for Atlas resulting in the hand resetting, which would leave the hand in an undesired state.

Due to the strength and reliability limitations of the actuated hands, we opted to use a fixed metal hook whenever possible. The hook was small, easy to use, and highly durable. It was a highly reliable end effector when actual grasping was not required.

3.4. User Interface

We developed a highly effective user interface (UI) for the VRC using a Coactive Design approach (Johnson, Bradshaw, Feltovich, et al., 2014). Coactive Design breaks with traditional autonomy-centered approaches by focusing on effective management of the interdependencies among human-machine team members (Johnson et al., 2011). Providing support for interdependence enables members of a human-machine team to recognize problems and adapt. Support for a variety of interdependence relations makes a team flexible. Flexibility, in turn, makes the team resilient by providing alternative ways to recognize and handle unexpected situations.

The Coactive Design method (Johnson, Bradshaw, Feltovich, et al., 2014) was the process used for both the design of new capabilities and analysis of existing ones. The approach was particularly useful in the redesign process necessary for transitioning to real-world operation using physical hardware, where assumptions needed to change and an understanding of the impact of those changes was essential to adapting both the interface and the underlying algorithms.

The next sections describe some of the modifications required for our user interface. Some modifications were the addition of new capabilities to handle the increased task complexity in the DRC Trials. The purpose of other modifications was to address the different context provided by the transition from simulation to real hardware.

3.4.1. Recognition of Small Features

The DRC Trials manipulation tasks involved smaller features than those of the VRC. It included things like door handles, hose coupling rings, and on/off buttons on drills. These required much finer grained resolution of information. In the VRC, LIDAR data were displayed using an Octree implementation (Koolen et al., 2013). However, this did not provide enough resolution required for new manipulation requirements. To support these cases, we included a near-field high-resolution mode. This mode provided the highest possible scan resolution within an adjustable radius around the robot while still maintaining Octree information outside the high-resolution area.

3.4.1.1. Coloring and Texture

Even with the higher-resolution LIDAR data, it was still difficult to distinguish small features, such as door handles and buttons. To address this, we employed two different techniques. The first was a coloring technique that changed the color of the point cloud elements as a function of the horizontal radius of the point cloud elements to the LIDAR sensor head, shown on the right side of Figure 8. This provided adequate information for the operator to differentiate most objects. The second technique was a texturing method, which allowed point cloud elements to adopt the color from the real object, shown in the center of Figure 8. This made point cloud interpretation easier for the operator and aided in identifying small features.

3.4.2. Addressing the Need for a Wider Field of View

During the VRC, we were able to successfully tackle most tasks using only the cameras and LIDAR from the Atlas sensor head. These were displayed to the operator through the First- and Third-Person-View displays of the user interface as described in Koolen et al. (2013). During the Trials, it became apparent that certain tasks, such as the door task and the debris task, would require a wider field of view than was afforded by the MultiSense-SL cameras. To support this, we integrated the BackFly cameras located on the robot's chest (Figure 9). These cameras are equipped with a fisheye lens and provide about 185 ° coverage each along the robot's side, compensating for Atlas' lack of neck yaw. When combined, both BlackFly cameras provide a full panoramic coverage of the robot's surroundings. Instead of

[8]Sandia hand: http://www.sandia.gov/research/robotics/advanced_manipulation/Sandia_Hand.html (accessed 18 SEP 2014).

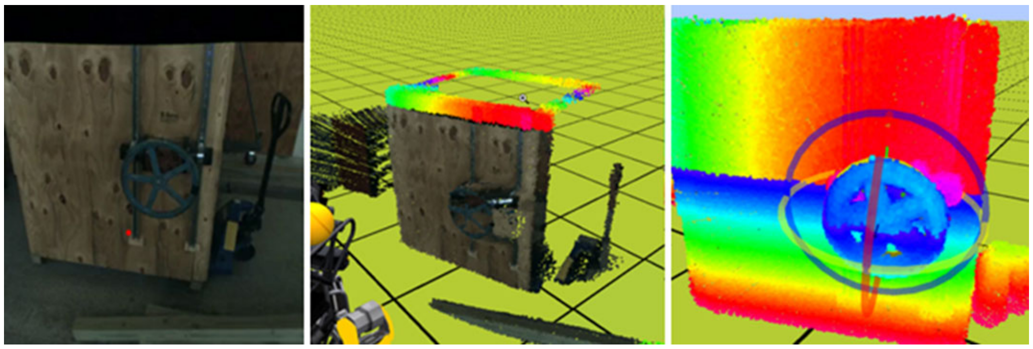

Figure 8. (Left) The actual camera image from the robot's camera as seen in the user interface. (Center) Colorized point cloud data using pixels from camera image. (Right) Colorized point cloud data based on depth of the point cloud data.

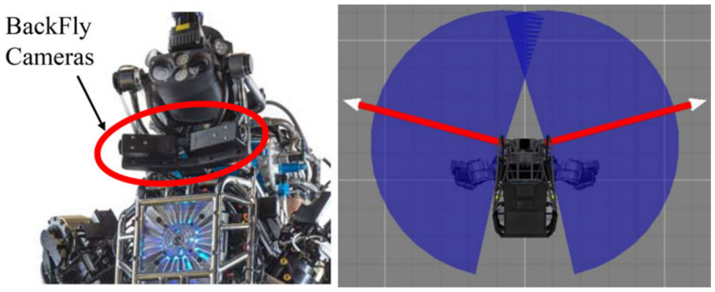

Figure 9. (Left) Location of the BlackFly cameras on Atlas. (Right) an overhead representation of the camera's field of view on the simulated robot.

displaying images from each camera (head and chest) on different windows within the UI, we fused the video images from all cameras together into the same viewport (Figure 10). This technique consists of projecting the images from each camera onto a virtual dome within a 3D environment. It effectively transforms the typical 2D camera view into a fully interactive 3D view that only uses a single window. This technique provides richer situation awareness information to the operator in a simplified manner. The operator could "look around" even though the Atlas has no ability to yaw its head. This provided a critical awareness of context outside the field of view of the normal camera.

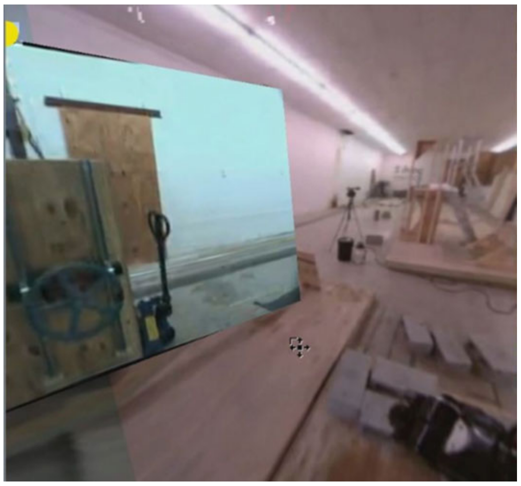

Figure 10. Images from head and chest cameras fused together into the first person viewport. The higher-resolution rectangle is from the normal camera, and the rest of the image is from the fisheye camera. This gives the operator the illusion of looking to the right even though the robot cannot yaw its head.

3.4.3. Understanding Joint Limits

The tasks of the DRC Trials pushed the Atlas to its limits. Atlas was frequently operated at its joint limits during the Trial tasks. Because of this, the operator would occasionally experience automation surprises (Woods & Sarter, 1997) that resulted in poor performance or robot failure. To help the operator better understand where the robot was within its operational envelope, we provided color indicators (Figure 11). They indicated whether a particular joint is close to or has reached a joint limit. This allowed the operator to better understand constraints and make more informed decisions. It also helped reduce automation surprises since the operator was aware of the constraints under which the automation was working.

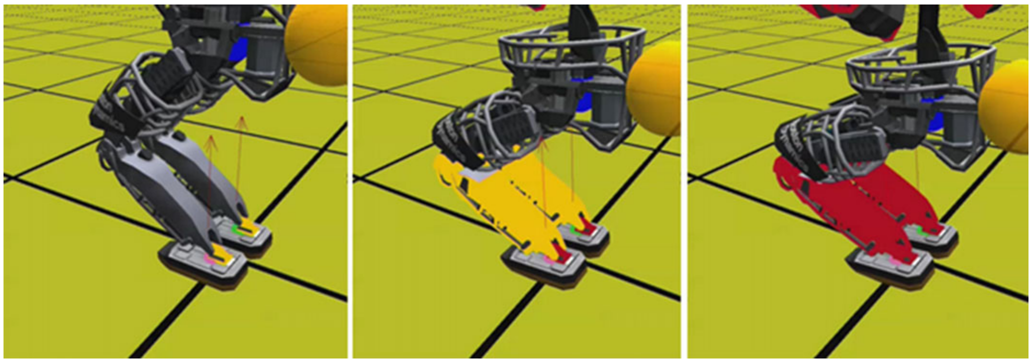

Figure 11. Color indicators used to show that joints are reaching their physical limits. (Left) Ankle within 10% of limit. (Center) Knee within 10% of limit and ankle at limit. (Right) Both knee and ankle at limit.

3.4.4. Safer Operation through Preview

When working with a large and powerful robotic system, safety must be the primary concern. In simulation, a bad arm motion had little impact. However, when Atlas is holding a powered cutting tool that is engaged, a bad arm motion can cause great damage to the surroundings or Atlas itself. Because of this, we elevated the information needs of the operator from simply seeing the desired position to being able to predict the complete motion required to achieve that end state. We added a trajectory visualizer that allows the operator see an animation of the arm moving through a trajectory in the workspace toward the desired position and orientation. This is depicted in Figure 12. This preview capability gave the operator the confidence to work aggressively in dangerous situations.

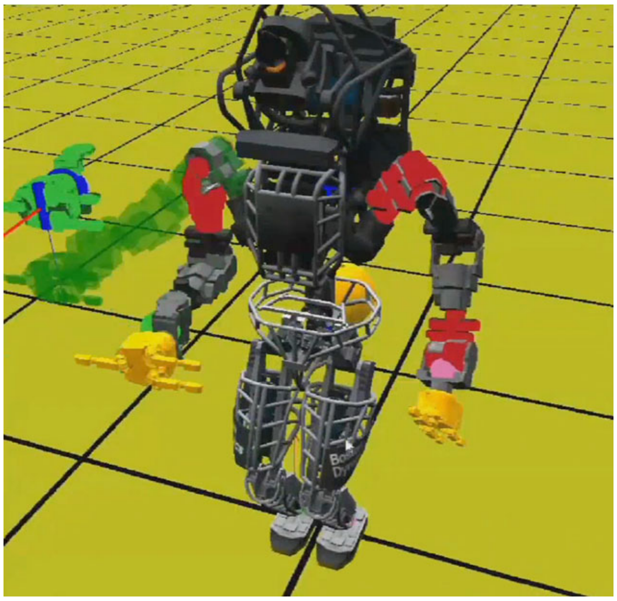

Figure 12. Motion preview capability providing the operator with predictability of the arm motion. This was critical to safe operation of a robot using power tools. The green hand is the desired end state. The green virtual arm is an animation that provides a preview of the motion that will achieve the end state.

3.4.5. Enabling the Operator to Specify More Complex Behavior

The DRC trials introduced many new manipulation tasks beyond what was needed for the VRC. In the VRC, we introduced *interactable objects* as an easy means for an operator to control the behavior of a complex high degree of freedom robot. These interactable objects are graphical components that the operator can place in the interface and manipulate as desired. For example, instead of manually placing a hand on a valve, the operator would place a valve interactable object and ask it for the appropriate hand position. As we moved toward more complex behavior requirements of the DRC Trials, we extended this concept to accommodate multistep behaviors as well as multioption behaviors. For example, our door interactable object controls not only approaching the door, but also opening it and passing through the door as well. It also provides multiple ways to approach the door. Additionally, the interactable objects provide more flexible control strategies. For example, instead of requiring the operator to specify a hand trajectory that generates a circular motion required to turn a valve, the interactable objects for the hand can be linked to the valve, and the trajectory is generated by simply turning the valve graphic. The traditional approach took more time and was less precise. The linked interactable object approach proved to be faster, safer, and more effective than the traditional approach. These linked relationships are generated on the fly by the operator and could be modified or removed at any time. Interactable objects are modeled to be as generic as possible, allowing them to be flexibly applied and easily adapted to different situations.

4. LESSONS LEARNED

In competitions like the DRC, the types of lessons learned are quite different from traditional scientific experimentation. These lessons are based on the experiences of IHMC's team in the competition, and our main evaluation criterion is overall performance against top notch competitors. Simply put, this is our view of what worked and what did not work, and these lessons should be interpreted in that context. The lessons can be grouped into the following categories: hardware, robot control, human-robot integration, software practices, and design choices.

4.1. Hardware Lessons

4.1.1. Solid, Reliable Hardware is Essential

We would not have been as effective in the competition if Boston Dynamics had not provided a robust and reliable platform. We intensively tested and developed on the Atlas for a period of five months with no maintenance issues. That is truly impressive. We eventually wore out some valves, but Boston Dynamics put in the extra effort required to ensure we were ready to compete. Developing on fragile hardware is frustrating and tedious, and our entry would have suffered if our robot had required much down time. Our experience with Atlas was quite positive. It is an example of the robustness and reliability systems will require as they move out of the lab into fielded systems.

4.1.2. The Need for Effective Hand Technology for Field Robotics

We had two different actuated hand options available, namely the Sandia hand and the iRobot hand, and we ended up using a basic hook for many tasks. We did this because of hardware reliability and robustness. Both hands have impressive videos demonstrating their manipulation abilities. However, our experience was that they lack sufficient strength when employed on a very strong and heavy hydraulic robot such as Atlas. Both hand technologies were too brittle for long-term field use. Though sufficient for many types of manipulation, fine-grained tasks, such as pushing the recessed button on a rotary tool,[9] remain challenging given the current state of hands. Developing hands that can provide fine-grained manipulation capability while being rugged enough and strong enough to work on robots like the Atlas remains an open challenge.

[9]Dewalt rotary tool used in competition: http://www.dewalt.com/tools/cordless-specialty-cordless-cut-out-tools-dc550ka.aspx (accessed on 18 SEP 2014).

4.2. Robot Control Lessons

4.2.1. Capturability-based Analysis and Control is an Effective Approach to Walking

For humanoids, walking is compulsory. Everything is contingent upon being able to reliably walk. Fielded humanoid robots are faced with the challenges of the imperfect world in which they must work. The ground is not always level, clutter can interfere with a footstep, and disturbances from the environment can affect the robot at any point during walking and manipulation. The complexity of walking is evidenced by only two of the 16 teams successfully completing the entire terrain task without requiring an intervention.

In our walking algorithm, we use a capturability-based control scheme (Pratt et al., 2012) to plan and control the momentum of the center of mass. We plan an instantaneous capture point trajectory based on the four upcoming footsteps, and we use this plan during real-time control. This method is an approximation since it is based on the assumption of a constant center-of-mass height. Deviations from a constant center-of-mass height, as well as parameter uncertainty and physical perturbations, will cause disturbances to the instantaneous capture point trajectory. These disturbances can be compensated for through feedback control, changing the Centroidal Moment Pivot. We find that this control scheme was highly effective for Atlas during the Trials, even when walking over cinder blocks where constant center-of-mass height was a poor assumption. One advantage of using an instantaneous capture point plan over a plan based on a full robot model is that it does not require planning of the trajectories of all of the degrees of freedom of the robot, but instead just plans the instantaneous capture point trajectory. Therefore, the plan is more flexible, and the motion of the limbs is not preconstrained during planning.

4.2.2. Real-time Control is Critical, and Existing Real-time Java Support is Insufficient

For robotic systems that are inherently unstable, such as a bipedal robot, real-time control is critical. IHMC's robotics group uses Java for all of their projects. However, none of the commercially available implementations of the Java Real Time Specification had the performance required to run our controller, requiring us to build our own solution. We feel Java has many advantages when developing large software code bases, but there remains a need for robust real-time support.

4.3. Human-robot Integration Lessons

4.3.1. Fielding Robotics Requires More than the Engineering Solution

The operator is a key part of the system. Designing good control algorithms and good software architectures is essential, but they will likely be dependent on effective use by an

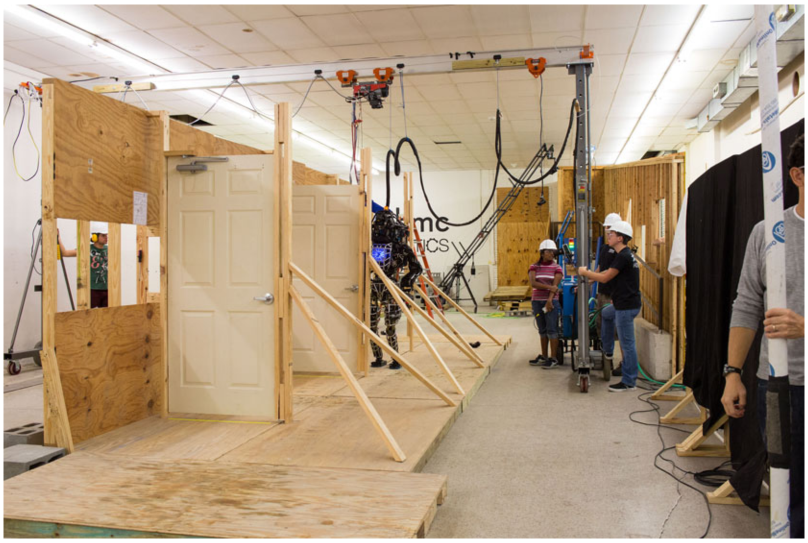

Figure 13. Example mock trial of the door task.

operator. This means that the operator must have an appropriate level of skill, must be provided an effective interface to leverage that skill, and must have adequate training. A large part of our success is due to a highly skilled operator who was well trained. We provided that operator with an effective interface designed in conjunction with the underlying control algorithms. We replicated the competition as best we could in our lab and ran repeated mock trials (see, e.g., Figure 13) to train our operators.

4.3.2. Coactive Design is an Effective Method for Designing for Human Involvement

It is not uncommon in robotics to use terms such as teamwork, collaboration, and human-in-the-loop. However, all these terms are too abstract to give direct guidance to human-robot system designers and developers. The challenge is to translate high-level concepts such as teamwork into specific requirements that can be implemented within control algorithms, interface elements, and behaviors.

When designing a human-robot system, there are some principles and guidelines (see, e.g., Goodrich, 2004; Goodrich & Olsen, 2003; Ponsa & Díaz, 2007) that are valuable, but they do not provide a design method. There are also examples of evaluations of existing system using these principles (see, e.g., Cummings, Nehme, Crandall, & Mitchell, 2007; Finn et al., 2012; Yanco et al., 2007), but these are evaluation methods, not design methods. There are some analysis methods such as hierarchical task analysis (Annett, 2003), cognitive task analysis (Crandall & Klein, 2006; Schraagen et al., 2009), and goal-directed task analysis (Endsley, Bolté, & Jones, 2003), which provide valuable insight into the needs of the human. Their main limitation is "focusing on the needs of the individual team members, often ignoring the collective decision making and coordination that is actually required (Cummings, da Silva, & Scott, 2007)." They also tend to consider only the human factors and the needs of the operator, and they ignore the engineering requirements and the needs of the robot. Additionally, guidance on designing human-machine systems often does not translate into the kind of specificity needed by engineers (Hoffman & Deal, 2008).

To address this need, we developed an approach to human-robot design that we call Coactive Design (Johnson et al., 2014). Coactive Design is the name of an approach and should be capitalized throughout this section. proved to be an effective way for our team to generate a specification usable by an engineer that effectively incorporates the human into the system. It uses the observability, predictability, and directability (Johnson et al., 2014) requirements of the human and robot to guide the design of not just the interface, but the algorithms and behaviors of the robot itself. In fact, you cannot effectively separate the two. Specific examples of how Coactive Design played a role in different design choices can be found in an upcoming article (Johnson, Bradshaw, Hoffman, Feltovich, & Woods, 2014). Coactive Design is unlike a traditional software specification in that it does not dictate what is required. Instead, it is more like a roadmap of runtime options, specifically options about how the human and robot might interact. The designer is free to choose which options to support based on cost/benefit, time, the desire for flexibility, or any of a myriad of factors. In the design phase, Coactive Design helped us to comprehend the constraints and opportunities as we tried to address eight new challenges for the DRC Trials. It also was invaluable in the redesign process, as it helped us to

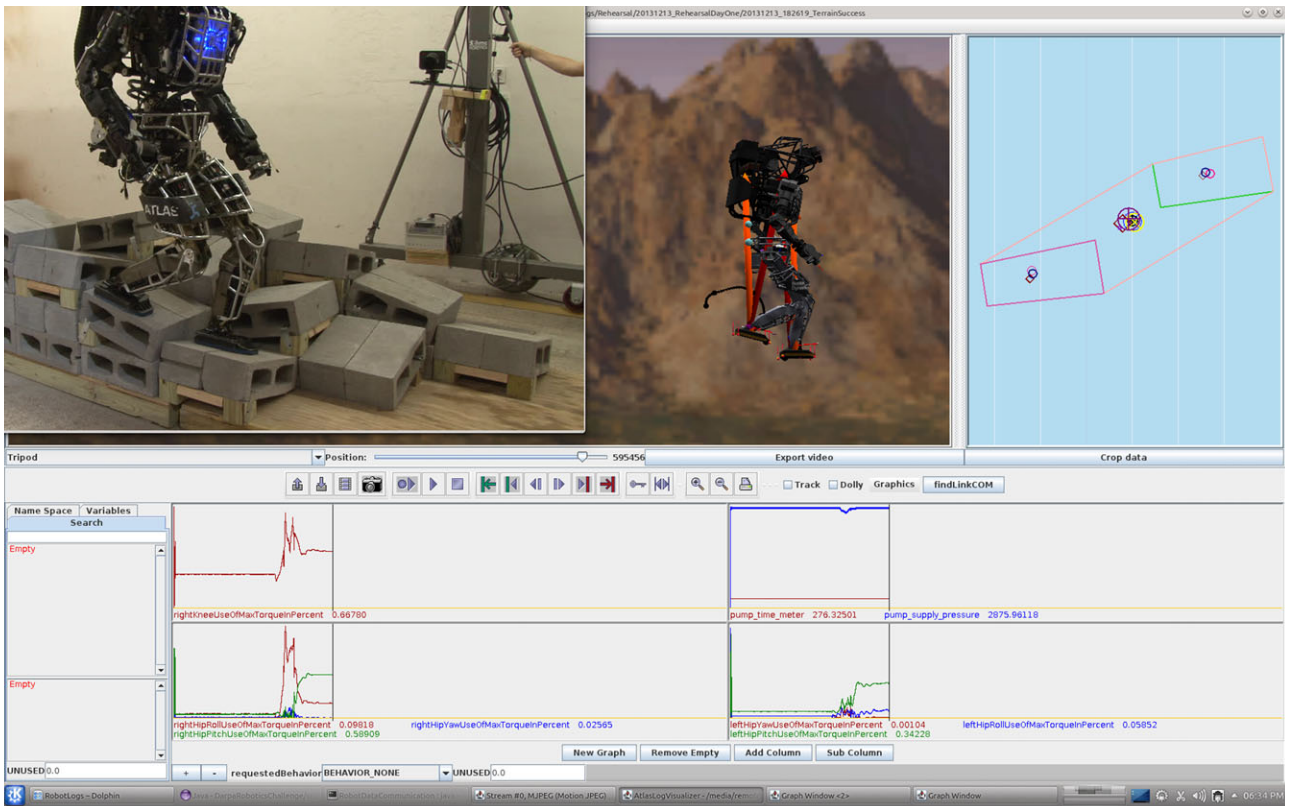
Tripod
Position:
595456
Export video
Crop data
Track
Dolly
Graphics
findLinkCOM
Name Space
Variables
Search
Empty
Empty
rightKneeUseOfMaxTorquePercent 0.66780
pump_time_meter 276.32501
pump_supply_pressure 2875.96118
rightHipRollUseOfMaxTorqueInPercent 0.09818
rightHipPitchUseOfMaxTorqueInPercent 0.58909
rightHipYawUseOfMaxTorqueInPercent 0.02565
leftHipYawUseOfMaxTorqueInPercent 0.00104
leftHipPitchUseOfMaxTorqueInPercent 0.34228
leftHipRollUseOfMaxTorqueInPercent 0.05852
New Graph
Remove Empty
Add Column
Sub Column
UNUSED 0.0
requestedBehavior BEHAVIOR_NONE
UNUSED 0.0

Figure 14. Logging tool displaying actual sensor data from the real robot in Simulation Construction Set and the synchronized video. This allowed developers to scrub data and video simultaneously. The logger runs automatically without the developer needing to start or stop the process.

understand how we need to modify both our algorithms and our interface to account for changes in moving from simulation to hardware. While the outward result is a unique interface, the actual work involved integration of the algorithms and interface in a way that allowed the operator to make effective use of their perception, judgment, and creativity in harmony with the underlying control algorithms.

4.4. Software Practice Lessons

4.4.1. Solid Software Practices are Worth the Extra Time

Effective companies know that software quality is critical. Schools teach this, but it is often difficult for people to buy into solid software practices, which come at a cost of time. It takes time to design first. It takes time to clean and refactor code. It takes time to write unit tests. Our experience has been that all of that time invested yields significant payback over the long term via reliable software and successful projects. We use a suite of software tools from Atlassian, our sponsor, to assist us in following good software practices.

4.4.2. Investing in Debugging Tools is Worthwhile

Evaluating failures of physical systems is a hard problem. Extreme programming using new hardware under deadlines is challenging. Failures happen so quickly, and it is often difficult to assess the cause. One of our strongest software assets when dealing with the physical Atlas is our logging suite. We invested time developing a logging tool (Figure 14) that is an extension of our simulation environment that the physical robot state is fed into the simulation along with all controller parameters and inputs. These data are used to display the simulated Atlas using the output from the physical Atlas' state estimator, and it can be shown in real-time. All of this information is time-stamped, logged, and merged with video from two dedicated high-definition lab cameras focused on Atlas. Developers could scrub these data along with the synchronized simulation and video. They could rewind the data, play them back in slow motion, or even step through them control tick by control tick. Analyzing these data allowed our team to identify whether failures were caused by software bugs, hardware problems, or misconceptions about the robot.

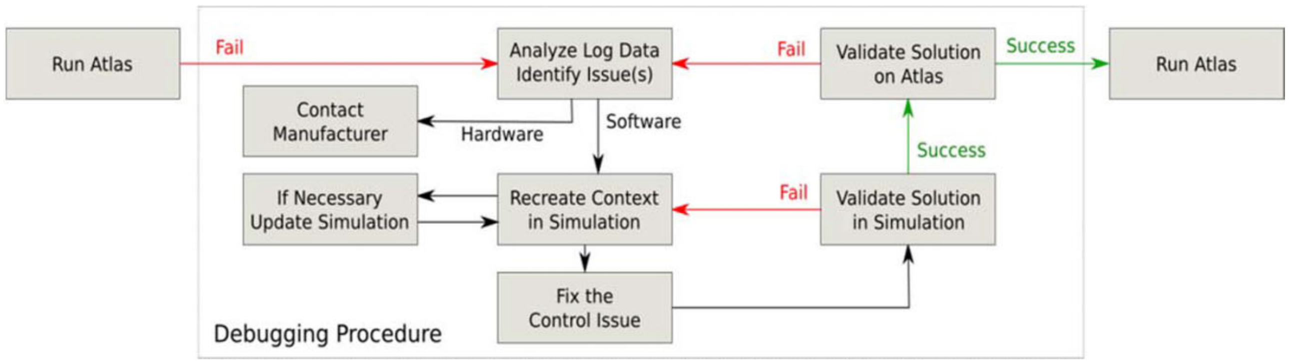
Run Atlas
Fail
Analyze Log Data Identify Issue(s)
Fail
Validate Solution on Atlas
Success
Run Atlas
Contact Manufacturer
Hardware
Software
Success
If Necessary Update Simulation
Recreate Context in Simulation
Fail
Validate Solution in Simulation
Fix the Control Issue
Debugging Procedure

Figure 15. IHMC testing procedure. The first step is to identify the issue and replicate it in simulation, then fix it in simulation, and finally validate the fix on the hardware. The software test now becomes a regression unit test to ensure future changes do not reintroduce this error.

4.4.3. A Suite of Unit Tests is the Only Way to Aggressively Develop with Confidence

The massive time constraints of the competition required an equally massive software development effort. Coding on such a scale will introduce many, many bugs. The only way not to be paralyzed by the fear of introducing bugs is to develop an intensive testing approach that gives you confidence that nothing has been broken. We used a tiered approach, using our existing simulation unit tests combined with black box testing of the physical robot, to help eliminate weaknesses in our software. Figure 15 shows our approach to dealing with failures. The simulation recreation fork allows us to investigate the limitations of the simulation and sources of error from the robot. In instances in which the simulation was unable to recreate the failure, we adapted the simulation to coincide with the real world as closely as possible, making it suitable for future unit test verification. Once the failure was introduced, a simulation test was created based on these conditions and the controller was refined until the test was passed along with the regression test suite. The final changes to the controller are validated using the physical Atlas. This may take a little longer than just fixing the problem, but the remaining software unit test will inform us if future software changes cause this to ever become an issue again. Our tests are divided into an every-commit test suite that runs in under 20 min, and a nightly test suite that runs numerous end-to-end acceptance tests of a simulated Atlas walking over multiple terrains while being disturbed by external forces. Nightly Atlassian builds that ran our suite of unit tests meant the source of error would be isolated to no more than 24 h of code.

4.5. Design Choice Lessons

4.5.1. Know When to Design for Sufficiency

It is important to differentiate when a criterion should be minimized and when only sufficiency is required. Bandwidth constraints are an example of this. Any approach must consider network limitations. Due to bandwidth constraints, a designer may be tempted to reduce bandwidth as much as possible. However, this is a sufficiency requirement, not a minimization requirement. Our view is that we should maximize the use of bandwidth as much as possible to fully utilize what is available. We did need to provide good compression algorithms and carefully design communication protocols to ensure we stayed under the limits (i.e., sufficiency), but our design goal was to provide the operator with as much relevant information as possible, since the operator is viewed as a critical teammate.

4.5.2. Human-Machine Teaming is a Viable Path to System Resilience

The primary metric that should be driving fielded systems is the likelihood of mission success, rather than technological innovation or theoretical elegance. For example, it is easy to become so focused on perfecting capabilities for machine autonomy that one forgets the fact that the performance goal can often be achieved more effectively and inexpensively by combining human and machine capabilities in better ways. DARPA has used many metrics during this competition, including task completion, speed, and bandwidth usage. From the beginning, we focused solely on mission completion with the understanding that it would be the deciding factor. This is particularly true given the complexity of tasks DARPA was proposing. We did need to complete the tasks in a given time, and we did need to adhere to DARPA's bandwidth limitations, but these were sufficiency requirements and not the driving force behind our work. In the end, mission completion was the most critical, as evidenced by DARPA using it as the primary differentiator of teams during the DRC Trials.

An essential and much neglected factor in mission performance is engineering for resilience. Resilience is not about optimal behavior; rather, it is about survival and mission completion. Resilience is the ability to recover from or adjust easily to misfortune or change.[10] David Woods describes it this way: "Resilience then concerns the ability to recognize and adapt to handle unanticipated perturbations that call into question the model of competence, and demand a shift of processes, strategies and coordination" (2006, p. 22). His description captures the two essential components of resilience: recognition of problems and flexible alternatives to address them.

The traditional approach to resilience is to improve "autonomy" — but such thinking, when applied to actual practice, falls prey to a series of deadly myths (Bradshaw, Hoffman, Johnson, & Woods, 2013). Engineering better autonomous capabilities is undeniably valuable. However, autonomy is often fragile due to its deficiencies in anticipating and recognizing problems that may inhibit mission success (Feltovich, Bradshaw, Clancey, Johnson, & Bunch, 2008; Woods & Branlat, 2010) and its inability to generate flexible alternatives to address them (Norman, 1990; Woods & Branlat, 2010). As evidence, even though the challenges in both the VRC and the DRC Trials were specified prior to the competition, there were *no* fully autonomous solutions to *any* of the tasks. Moreover, previous robot competitions have noted that "the more complex phases of the challenge naturally stress the capabilities of the robots more, which in turn leads to increased human intervention" (Finn et al., 2012).

We propose human-machine teaming as an alternative approach to achieving resilience. However, simply putting a human "in-the-loop" does not guarantee effective human-machine teaming. If teaming is to be a viable alternative approach to resilience, then it is important to understand how a developer designs a system to work effectively as a teammate. Coactive Design provided our team with this understanding and enabled us to build flexibility and thereby resilience into our system. Examples of resilient behavior were more obvious in the VRC where we were required to perform each task multiple times (see Johnson et al., 2014), but there were also instances during the DRC Trials (e.g., the wind closing doors during the door task). In both cases, our success was not because we performed flawlessly, but instead reflects our system's resilience to recover from errors and unanticipated circumstances and adapt to overcome them.

5. CONCLUSION

In a short period of time, team IHMC created a working system that was capable of completing several challenging disaster response tasks in a complex environment using tools and equipment designed for human use and under degraded communications as part of the DARPA Robotics Challenge (DRC). In the Virtual Robotics Challenge, IHMC was the top scoring team, and in the DRC it was the second highest scoring team. This success was brought about through a combination of capable robot hardware, innovative walking control algorithms, effective human-robot teaming using the Coactive Design method, diligent software development practices, and maximizing our limited resources using appropriate design choices. Without any of these key factors, the outcome would likely have been very different.

[10]"Resilience." Merriam-Webster.com. Merriam-Webster, n.d. Web. 1 Feb. 2014. http://www.merriam-webster.com/dictionary/resilience.

ACKNOWLEDGMENTS

We would like to thank DARPA for sponsoring the Robotics Challenge and encouraging the advancement of robotics capabilities. We would also like to thank DARPA for the funding provided to IHMC to compete in the competition. We must also thank Boston Dynamics for providing Atlas, which has been a solid and reliable robotic platform. They also went above and beyond expectations by working very late to ensure our robot was ready to perform at the DRC Trials. Lastly, we would like to acknowledge Atlassian, our sponsor, who provided an embedded engineer to ensure our agile practices were effectively applied using their Atlassian software tools.

REFERENCES

Bradshaw, J. M., Hoffman, R. R., Johnson, M., & Woods, D. D. (2013). The seven deadly myths of "autonomous systems." IEEE Intelligent Systems, 28(3), 54–61.

Cummings, M. L., da Silva, F. B., & Scott, S. D. (2007). Design methodology for unmanned aerial vehicle (UAV) team coordination. MIT Humans and Automation Laboratory Report.

Cummings, M. L., Nehme, C. E., Crandall, J., & Mitchell, P. (2007). Predicting operator capacity for supervisory control of multiple UAVs. In Innovations in intelligent machines (pp. 11–37). Springer.

Feltovich, P. J., Bradshaw, J. M., Clancey, W. J., Johnson, M., & Bunch, L. (2008). Progress appraisal as a challenging element of coordination in human and machine joint activity. In A. Artikis, G. M. P. O'Hare, K. Stathis, & G. Vouros (eds.), Engineering societies in the agents world VIII (pp. 124–141). Heidelberg, Germany: Springer.

Finn, A., Jacoff, A., Del Rose, M., Kania, B., Overholt, J., Silva, U., & Bornstein, J. (2012). Evaluating autonomous ground-robots. Journal of Field Robotics, 29(5), 689–706.

Goodrich, M. A. (2004). Using models of cognition in HRI evaluation and design. Defense Technical Information Center.

Goodrich, M. A., & Olsen, D. R. (2003). Seven principles of efficient human robot interaction. In IEEE International Conference on Systems, Man and Cybernetics (Vol. 4, pp. 3942–3948).

Johnson, M., Bradshaw, J. M., Feltovich, P. J., Jonker, C. M., van Riemsdijk, M. B., & Sierhuis, M. (2014). Coactive design:

Designing support for interdependence in joint activity. Journal of Human-Robot Interaction, 3(1), 43–69.

Johnson, M., Bradshaw, J. M., Hoffman, R. R., Feltovich, P. J., & Woods, D. D. (2014). Seven cardinal virtues for human-machine teamwork: Examples from the DARPA Robotics Challenge. IEEE Intelligent Systems, (6), 74–80.

Koolen, T., Smith, J., Thomas, G., Bertrand, S., Carff, J., Mertins, N., & Pratt, J. E. (2013). Summary of team IHMC's virtual robotics challenge entry. In Proceedings of the IEEE-RAS International Conference on Humanoid Robots. Atlanta: IEEE.

Norman, D. A. (1990). The "problem" of automation: Inappropriate feedback and interaction, not "over-automation." In D. E. Broadbent, A. Baddeley, & J. T. Reason (eds.), Human factors in hazardous situations (pp. 585–593). Oxford University Press.

Odhner, L. U., Jentoft, L. P., Claffee, M. R., Corson, N., Tenzer, Y., Ma, R. R., & Dollar, A. M. (2013). A compliant, underactuated hand for robust manipulation. Robotics. Retrieved from http://arxiv.org/abs/1301.4394

Ponsa, P., & Díaz, M. (2007). Creation of an ergonomic guideline for supervisory control interface design. In Proceedings of the 7th International Conference on Engineering Psychology and Cognitive Ergonomics (pp. 137–146). Berlin, Heidelberg: Springer-Verlag. Retrieved from http://dl.acm.org/citation.cfm?id=1784197.1784213

Pradeep, V., Konolige, K., & Berger, E., (2014, January). Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach. In Experimental Robotics (pp. 211–225). Berlin, Heidelberg: Springer.

Pratt, J., Koolen, T., de Boer, T., Rebula, J., Cotton, S., Carff, J., & Neuhaus, P. (2012). Capturability-based analysis and control of legged locomotion, Part 2: Application to M2V2, a lower body humanoid. International Journal of Robotics Research, 31(10), 1117–1133.

Woods, D. D., & Branlat, M. (2010). Basic patterns in how adaptive systems fail. In J. Pariès, E. Hollnagel, J. Wreathall, & D. D. Woods (eds.), Resilience engineering in practice: A guidebook (pp. 127–144). Ashgate.

Woods, D. D., & Hollnagel, E. (2006). Joint cognitive systems: Patterns in cognitive systems engineering. Taylor & Francis.

Woods, D. D., & Sarter, N. B. (1997). Automation surprises. In G. Salvendy (ed.), Handbook of human factors & ergonomics (2nd ed.). Wiley.

Yanco, H. A., Keyes, B., Drury, J. L., Nielsen, C. W., Few, D. A., & Bruemmer, D. J. (2007). Evolving interface design for robot search tasks. Journal of Field Robotics, 24(8-9), 779–799.