Achieving Reliable Humanoid Robot Operations in the DARPA Robotics Challenge: Team WPI-CMU's Approach

Christopher G. Atkeson[*], Benzun P.W. Babu[†], Nandan Banerjee[†], Dmitry Berenson[†],

Christoper P. Bove[†], Xiongyi Cui[†], Mathew DeDonato[†], Ruixiang Du[†], Siyuan Feng[*],

Perry Franklin[†], Michael A. Gennert[†], Joshua P. Graff[†], Peng He[†], Aaron Jaeger[†],

Joohyung Kim, Kevin Knoedler[†], Lening Li[†], Chenggang Liu[*], Xianchao Long[†],

Felipe Polido[†], X Xinjilefu[*], Taşkın Padır[‡],

[*] Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania
[†] Robotics Engineering Program, Worcester Polytechnic Institute, Worcester, Massachusetts
[‡] Department of Electrical and Computer Engineering, Northeastern University, Boston, Massachusetts

1 Introduction

The DARPA Robotics Challenge (DRC) was aimed at responding to natural and man-made disasters by human-robot teams (Pratt and Manzo, 2013). What if we had been able to prevent hydrogen explosions in the Fukushima Daiichi Nuclear Power Plant by using robots within the first hour after it was hit by a tsunami triggered by the Great East Japan earthquake in 2011? Since its announcement in 2012, the DRC has mobilized hundreds of robotics researchers, practitioners and makers to accelerate the research and development in robotics for disaster response. The DRC Finals on June 5-6, 2015 brought 23 qualified teams to Pomona, CA to demonstrate their systems in a disaster mission scenario. In a simulated environment, robots performed a variety of manipulation and mobility tasks under human supervision with a 1-hour mission completion time. For greater realism, the communications between the operator(s) and robot was degraded during parts of the mission.

Humanoid robots have advantages for completing a wide variety of tasks in environments sized and shaped for humans, such as traversing doorways, hallways, and stairs, manipulating door handles and other controls such as valve knobs and handles, using tools designed for humans, and driving a vehicle. However, despite receiving great attention, humanoid robot motion planning and control remain challenging research topics. Completion of the DRC mission with a humanoid robot required the development of reliable and accurate techniques for perception, full-body motion planning and control, dexterous manipulation as well as easy-to-use and intuitive operator interfaces.

The paper is organized as follows: We describe our approach in the DRC Finals including brief descriptions of our hardware and software systems, optimal control for walking and manipulation, optimization-based motion planning, perception, and our validation strategy. We then present results, and lessons learned.

Figure 1: The ATLAS Unplugged humanoid robot developed by Boston Dynamics.

2 Approach

2.1 Robot

Atlas Unplugged. At the DRC Finals, Team WPI-CMU competed with ATLAS Unplugged. The Boston Dynamics' Atlas Unplugged humanoid robot has 30 Degrees of Freedom (DoF), weighs approximately 180Kg, and is 1.8 meters tall (Figure 1). Its lower limbs, torso, and upper arms are composed of hydraulic joints, while the forearms and wrists are electrically powered. All joints have position and force sensing, and there is a high quality inertial measurement unit (IMU) in the pelvis. The robot also has a Carnegie Robotics' Multisense SL featuring a stereo camera and a spinning Hokuyo UTM-30LX-EW LIDAR for perception, as well as additional cameras on the head. The robot includes a battery system and wireless communication to operate without a tether. The unplugged upgrade also moved the cooling, computing and power conversion onto the robot. This along with the battery backpack allows for completely "unplugged" operation of the robot.

Our robot, WARNER(WPI's Atlas Robot for Non-conventional Emergency Response), is also equipped with two Robotiq 3-finger hands as end-effectors, video cameras on the wrists (looking at the hands) and knees (looking at the feet), and multiple MEMs-based IMUs on various links to improve state estimation.

2.2 Computational Resources

On-board Computers. ATLAS Unplugged is equipped with 3 onboard Pentium i7 computers that were directly connected to each other through 3-gigabit Ethernet networks. On top of the direct network connections the 3 computers are also connected through a network switch that also connects to the wireless

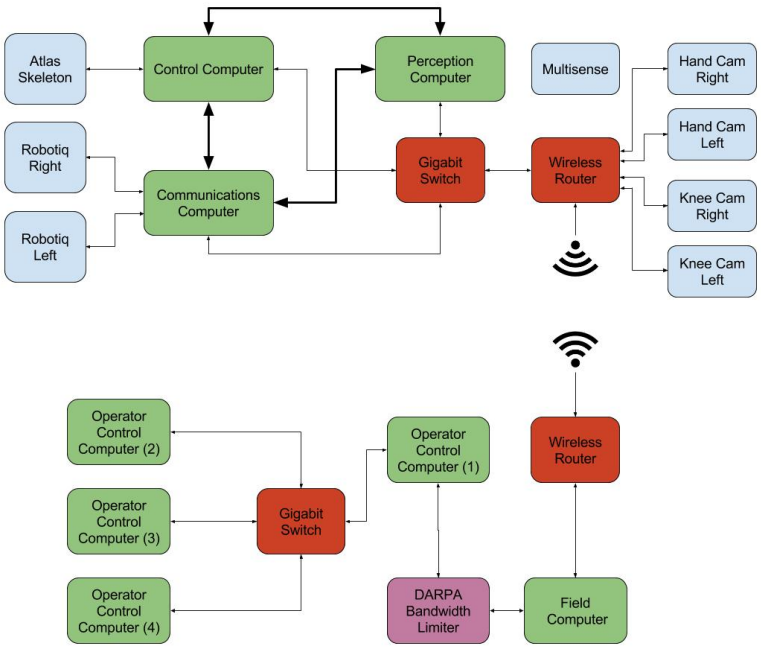

Figure 2: A visualization of the computational resources and network connectivity on board ATLAS Unplugged.

network. Each of the 3 computers had dedicated tasks assigned to them, and would send the data through the appropriate network based on the shortest path. The control computer connects directly to the robot through a separate network interface. This computer is dedicated to balancing and motion control algorithms, and runs the low level joint control in a tight 1kHz loop. The vision computer runs all of the drivers and algorithms for the cameras and computer vision. A Carnegie Robotics Multi-Sense head is connected directly to this computer through a dedicated network port. This computer handles the added hand and foot cameras that are connected to the gigabit switch. The last computer handles communication to the operator over the wireless network and control of the two-Robotiq hands that are connected directly to two dedicated network ports. Since this computer had no time critical, or processor intensive tasks, it also took care of all of the high-level task control. On the other side of the wireless link is the field computer.

Field Computer. In our implementation, the field computer only has the basic task of converting the TCP packets from the wireless link, to UDP packets that are sent over the limited bandwidth connection. The operator control station consists of 4 operator control units. The main operator control unit converts packets coming from the limited bandwidth connection back into ROS messages. The ROS messages are then sent to the GUI's that run on each of the operator control units.

2.3 Software Architecture

Our software architecture design has been driven by the goal to enable fast human-in-the-loop control of a humanoid robot over limited or degraded links (Figure 3). The degraded communications links are handled in a way so that user intervention is not needed as the link quality changes. The communications links were degraded such that 9600 bits/second was always available in each direction (Figure 3). One second bursts of 300 Mbits/second were available from the robot to the operator with 0-30 second dropouts dependent on the location of the robot and the overall elapsed time. The 9600 bps links had to be managed so that no frequent or large packets were sent across them. Control messages from the operator to the robot were

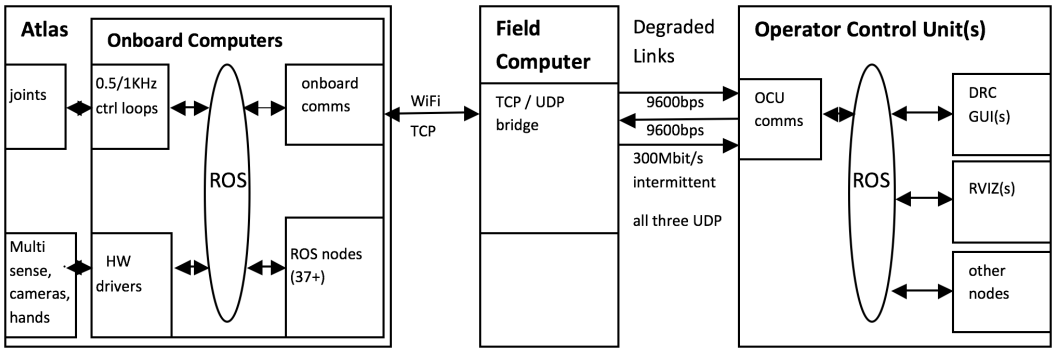

Figure 3: Both the WiFi and the link between the Field Computer and the OCUs were limited.

generally not a problem as the operator did not generate large or high frequency data sets. Status from the robot to the operator was much more of a problem. A single packet has about 50 bytes of headers. A single process could easily send status at 500 Hz and overwhelm the 9600 bps link resulting in high latency and loss of other data on that link. To mitigate this each of the channels within the link was limited to 3 Hz by default. This worked for most channels, but a few channels required every message to get through - for example state machine transitions. In addition, those transitions could happen in less than 300 mS per transition. So, state machine transitions had to be handled differently than other messages. A small image or data set (10 Kbyte) would take roughly 10 seconds to be sent across the link. Most data was kept small so that the 9600 bps link to the operator was not filled with old data. The overall robot status (joint positions/force, battery, pose, mode, etc.) was compressed to a single 200 byte packet and sent regularly to allow other intermittent data such as manipulation or footstep plans to be sent as needed.

The 300 Mbit/second intermittent link was used for sending large data sets such as depth maps, point clouds and images when the link was available. The robot pose data from the slower link was merged in with the last point cloud to provide an updated position of the robot relative to the environment as the robot moved during the dropouts. Both the position of the robot and joint changes were updated on the point cloud. This allowed the operator to continue monitoring and/or operating even during the dropouts that lasted up to 30 seconds.

2.4 Operator Interfaces

A guiding principle of our operator interface design was to provide the operator with the ability to control tasks from a very high level down to the joint level (Figure 4). Rather than a completely different interface for every task, each task interface had a considerable degree of commonality. Our interface supported different types of human inputs that applied to all tasks, as well as task-specific inputs. Operators could command tasks or task components, and specify movements or velocities in joint coordinates, Cartesian coordinates, and task coordinates. The ability to directly teleoperate the robot and the ability to command the hands to do simple grasping operations were used to perform the surprise tasks. Stored behaviors could be replayed. Targets could be designated on displayed images and point clouds.

Allowing the operator to control and monitor the robot at varying degrees of abstraction is important for handling unexpected situations. If everything goes according to plan tasks can be handled as simply as clicking next as the robot autonomously proceeds through the subtasks for completing a specific task. For example in the valve task, once the perception system detects the approach vector to the valve, and operator validates it, the robot engages the valve and opens it. These scripted subtasks are validated experimentally during the development. When the unexpected happens the operator needs the ability to do more than just click next. During the trials in 2013 this allowed recovery when the robot's foot slipped off an accelerator. During the finals this allowed recovery when the robot got stuck on the door frame, when a wrist joint

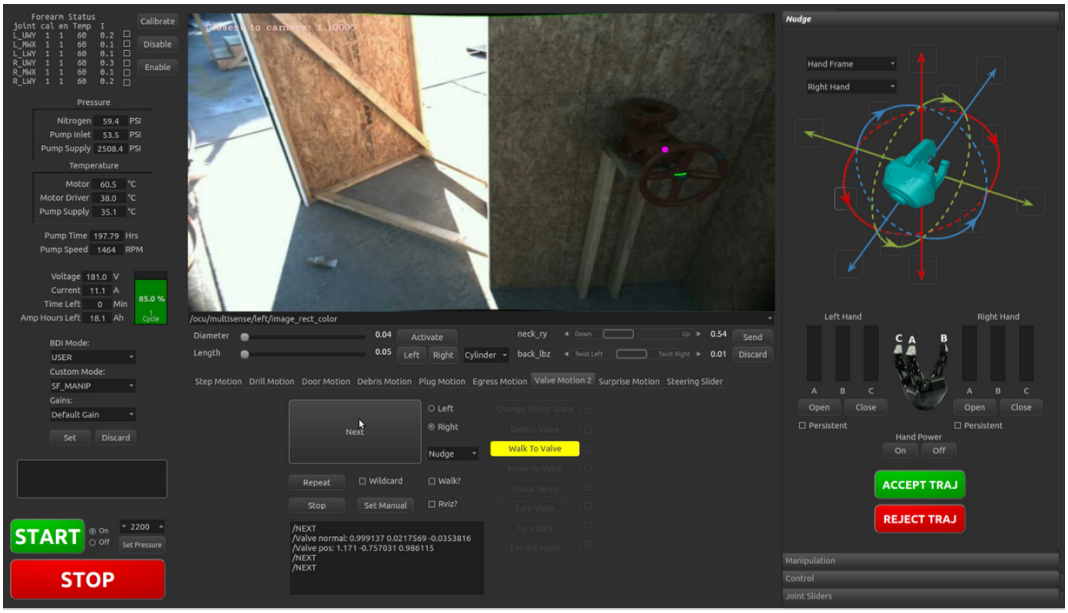

Figure 4: Team WPI-CMU's graphical operator interfaces allowed for task customization even though there were many commonalities from task to task.

overheated, and when a footstep landed such that it was on two blocks of differing z height. In these cases middle and lower level interfaces allowed the operator to control the robot and recover from the unexpected situation.

When something unexpected happens, the experience of an operator who knows the task and software in detail is needed. We used separate trained operators for different tasks. Replicated (or parallel) operator control units (OCUs) are provided to allow fast operator switch-over. At the DRC Trials in 2013, each task was allocated up to 30 minutes to complete with a large break between tasks. This allowed getting the required software in place and allowed the operators for that task time to set up. At the finals in 2015, all 8 tasks had to be completed in 60 minutes. The architecture provided four parallel OCUs so that operators could get any desired interfaces prepared before their task and be ready to operate the robot as soon as the previous task was completed. One challenge this posed was keeping the user interfaces consistent. While one OCU was being used the others had to be updated with the inputs/state from the other OCUs. One way to think of the multiple OCUs is a pilot/co-pilot in an aircraft. Either the pilot or co-pilot could control the robot and either one could step away to allow the best pilot to control a particular task. In the same way the instruments and controls (throttle/rudder/stick/instruments) need to track one another so that the operator is not surprised as they take control.

Results: Error Detection and Recovery Using Our Interface: While walking over the rough terrain on day 1, there was a bad step, which was rapidly detected by human operators and the robot was stopped ("frozen"). The human operators developed a recovery strategy, and the robot successfully executed it without falling down. What we believe happened is that something shortened the step (perhaps the heel touched the ground during swing) and put the foot at an angle supported by two different cinder blocks on touchdown, rather than fully supported by one cinder block (Figure 5). We were impressed that the controller was able to handle this so that the robot had the opportunity to recover, and that our operators could recover the robot without letting it fall or require physical human intervention. Similar recoveries were achieved in other situations (Figures 6 and 8).

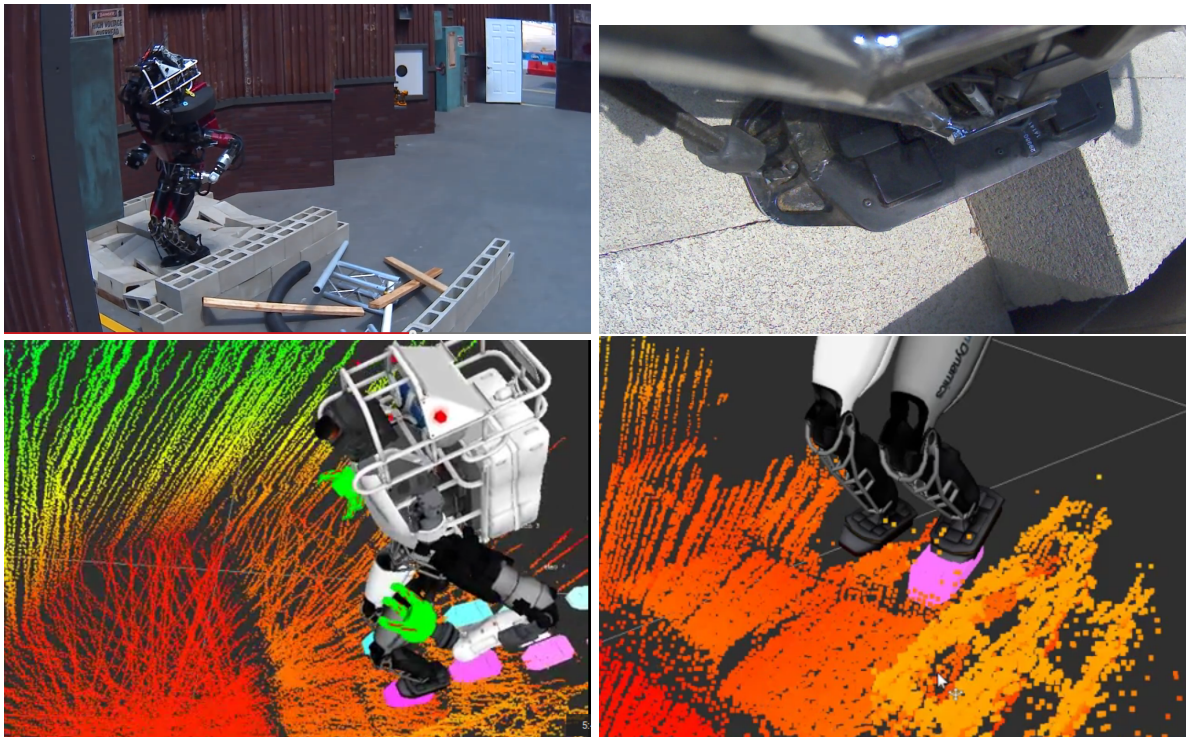

Figure 5: Terrain Error: Top Left: The "frozen" robot, Top Right: A view of the left foot from the knee camera showing it straddling two cinder blocks. Bottom row: Mismatch between the plan and the actual step

2.5 Optimal Control for Walking and Manipulation

We chose online optimization to generate behavior because it can easily generate a wide range of behaviors with grasp or footstep targets. This has been demonstrated convincingly throughout the DARPA Robotics Challenge considering the short development time and the large number of different tasks that needed to be completed. We developed optimization-based walking and manipulation controllers. Both consisted of a high-level controller that optimizes task space trajectories into the future and a low-level full-body controller that generates instantaneous joint level commands that best track those trajectories while satisfying physical constraints using Quadratic Programming (QP) (Feng et al., 2015). The high-level controllers output desired Cartesian motions for specific points on the robot (e.g. foot, hand, and CoM). The full-body controllers take these as inputs and generate joint level commands such as joint position, velocity, and torque, which are then used as desired values in the joint level controllers.

Since the DRC Trials at the end of 2013, we have improved our previous walking controller's implementation on the Atlas robot. We have also worked on tuning gains and filter parameters on the hardware for better joint level tracking performance. The low level full-body controller is also redesigned to address an inconsistency issue between the inverse kinematics (IK) and inverse dynamics (ID) modules due to their different sets of constraints encountered during the DRC Trials (DeDonato et al., 2015). With these changes, we have gained significant improvements in terms of performance and reliability. At top speed, our Atlas is able to walk over 20 times faster than in 2013. We also completed two one hour long missions at the Finals without any physical human intervention. Our Atlas can consistently walk over 30m in an outdoor parking lot as well.

Manipulation: For manipulation, inverse kinematics was used to generate the full state that best tracks the desired Cartesian commands. Inverse dynamics was then used to track the targets from inverse kinematics. We used both precalculated trajectories and TrajOpt (Schulman et al., 2013) to plan trajectories in real-time.

The details of our motion planning approach are discussed in Section 2.6.

Walking: Our approach was to plan footstep locations, and then have the controllers attempt to place the feet on those footstep locations. Early versions of our foot placement algorithms are described in (Huang et al., 2013) and the versions for the DRC Finals are described in (Feng et al., 2015). Future work will explore specifying cost functions for footstep locations and letting the optimization-based control trade off footstep error, timing, effort, and risk.

Given a sequence of desired footsteps, the walking controller optimizes a center of mass (CoM) trajectory using Differential Dynamic Programming (DDP), together with a local linear approximation to the optimal policy and a local quadratic approximation to the optimal value function, which are used to stabilize the robot around the nominal trajectory. DDP is a local iterative trajectory optimization technique using second order gradient descent that can be applied to nonlinear dynamics. A nonlinear point mass model that includes the vertical (Z) dimension is used for trajectory optimization to take height changes into account. The CoM trajectory is replanned during every single support phase for the next two footsteps. The swing foot trajectory is generated by a quintic spline from the liftoff pose to the desired touch down pose.

Step timing: Reliability was our primary objective for the DRC Finals, so a more quasi-static walking style was preferred. Having a slow cadence allowed us to pause walking at any point and gives the operator a chance to recover when things go wrong. In the DRC Finals, we used a nominal step length of 40cm and step time of 4s, leading to a stride period (a left step followed by a right step) of 8s. Among the top three Atlas teams we had the lowest cadence but took the longest foot steps. On the other hand, the controller is capable of more dynamic walking, and we have achieved 0.4m/s walking reliably by just speeding up the cadence (step time 0.8s, stride period 1.6s). We also note that of the top three Atlas teams, our footfalls seem to be the most gentle, and shock waves up the body on each foot fall are apparent in both IHMC's and MIT's walking. Does a firm footstep (a stomp) make sensing and control of locomotion easier or harder? This remains to be investigated in future research.

State Estimation: In addition to a state estimator for the pelvis (Xinjilefu et al., 2014), we also implemented an extended Kalman filter that estimates the modeling error at the CoM level using linear inverted pendulum (LIPM) dynamics (Xinjilefu et al., 2015). For the LIPM model, the modeling error can be treated as an external force or a center of mass offset. We treat it as an external force and compensate for it in the inverse dynamics controller. This estimator is especially helpful when compensating for unplanned slow changing external forces applied at unknown locations on the robot, which is quite likely when operating in tight spaces. It also handles relatively small dynamic forces well when walking, e.g. dragging a tether or pushing through a spring loaded door. Thanks to the estimator, very little tuning is done for our mass model.

Safety Code: Fall Prediction: The most significant contribution of the external force estimator is that it can detect when a large external force is being applied that might push the robot over. We compute a "corrected capture point" (CCP), which is an offset to the current capture point (Pratt et al., 2006). The offset takes into account the estimated external force, represented as an offset to the center of mass. The corrected capture point getting close to the boundary of the polygon of support warns the controller that the robot might fall if the external force is maintained. We can also compute the corrected capture point assuming that the external force follows a known time course plus an estimated constant offset, or steadily increases or decreases for a fixed time interval based on an estimated derivative. We assume the external force is due to the robot's actions, and not due to external disturbances such as wind, a moving support platform, or external agents pushing on the robot. When a fall is predicted, the current behavior is stopped and the robot "frozen" in place. This early warning system based on the corrected capture point saved our robot from falling twice at the DRC Finals, where no safety belay was allowed, and made us the only team that tried all tasks without falling and or physical human intervention (a reset).

A derivation of the corrected capture point starts with LIPM dynamics augmented with a true center of

Figure 6: Successful sidestepping through the door (left, middle) and the failure in the DRC rehearsal (right) in which the protective cage (black padding) for the head is against the white door frame.

mass offset and a true external force:

$$\ddot{c} = \left(c + c_{\text{offset}} + f_{\text{ext}} \frac{z}{mg} - COP \right) g/z = (c + \Delta - COP) g/z \tag{1}$$

where c is the location of the center of mass projected on the ground plane, COP is the center of pressure location in that ground plane, and Δ is the sum of the true center of mass offset from the modeled center of mass and any external horizontal force. Our extended Kalman filter estimates $\hat{c}$, $\hat{\dot{c}}$, and $\hat{\Delta}$, taking into account the current center of mass height z. We assume a constant center of mass height in estimating the corrected capture point based on the estimated capture point as described in (Pratt et al., 2006):

$$\widehat{CCP} = \widehat{CP} + \hat{\Delta} = \hat{c} + \hat{\dot{c}}\hat{z}/g + \hat{\Delta} \tag{2}$$

Predicting falling during walking is more complex (Xinjilefu et al., 2015).

Results of Safety Code: Robot caught on door frame: In the DRC rehearsal, the robot was caught on the door frame when sidestepping through (Figure 6). The walking controller detected an anomaly in the estimated external force in the sideways direction (F_x), delayed liftoff and remained in double support, and stopped the current behavior to allow for manual recovery (Figure 7).

Results of Safety Code: Manipulation Error: For the manipulation controller, the robot is always assumed to be in double support, and the support polygon is computed by finding the convex hull of the foot corner points (light green in Figure 8), computed using forward kinematics. To prevent the robot from falling during manipulation, we require the corrected capture point to be within a subset of the support polygon called the safe region, (dark green in Figure 8), When the corrected capture point escapes the safe region, a freeze signal is sent to the manipulation controller, and it clears all currently executing joint trajectories and freezes the robot at the current pose, with the balance controller still running.

During our second run in the Finals, our right electric forearm mechanically failed when the cutting motion was initiated for the drill task. The uncontrolled forearm wedged the drill into the wall and pushed the robot backwards. The controller stopped the behavior (a freeze), and saved the robot from falling (Figure 8). The operator was then able to recover from an otherwise catastrophic scenario.

The time plot in Figure 8 shows candidate fall predictors during this event. We can eliminate some candidate fall predictors easily. The center of mass (CoM) (and a "corrected" CoM (not shown)) usually provide a

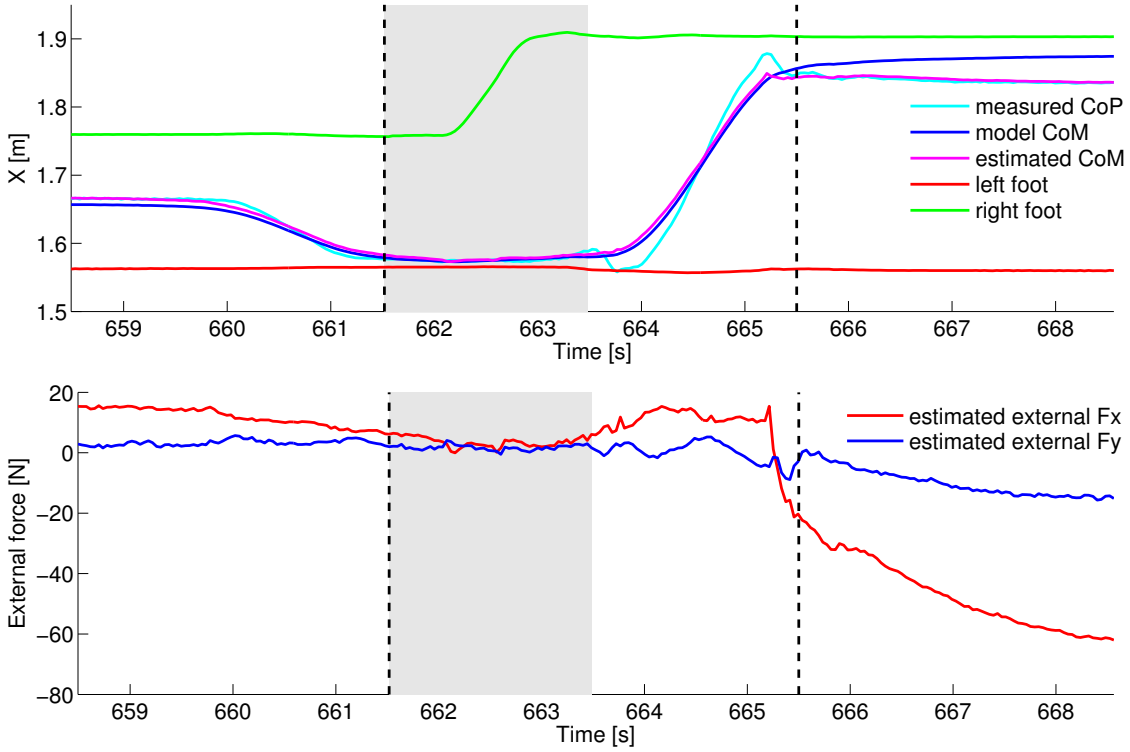

Figure 7: Atlas was caught on the door frame when sidestepping through it during the DRC rehearsal. The walking controller delayed liftoff and remained in double support when the external force estimator detected a large change in the estimated external force in the robot's sideways direction (F_x, through the door). The single support phase is shown by the shaded area, and the black dashed lines indicates the planned liftoff time. The estimated CoM is the sum of the model CoM and the estimated CoM offset.

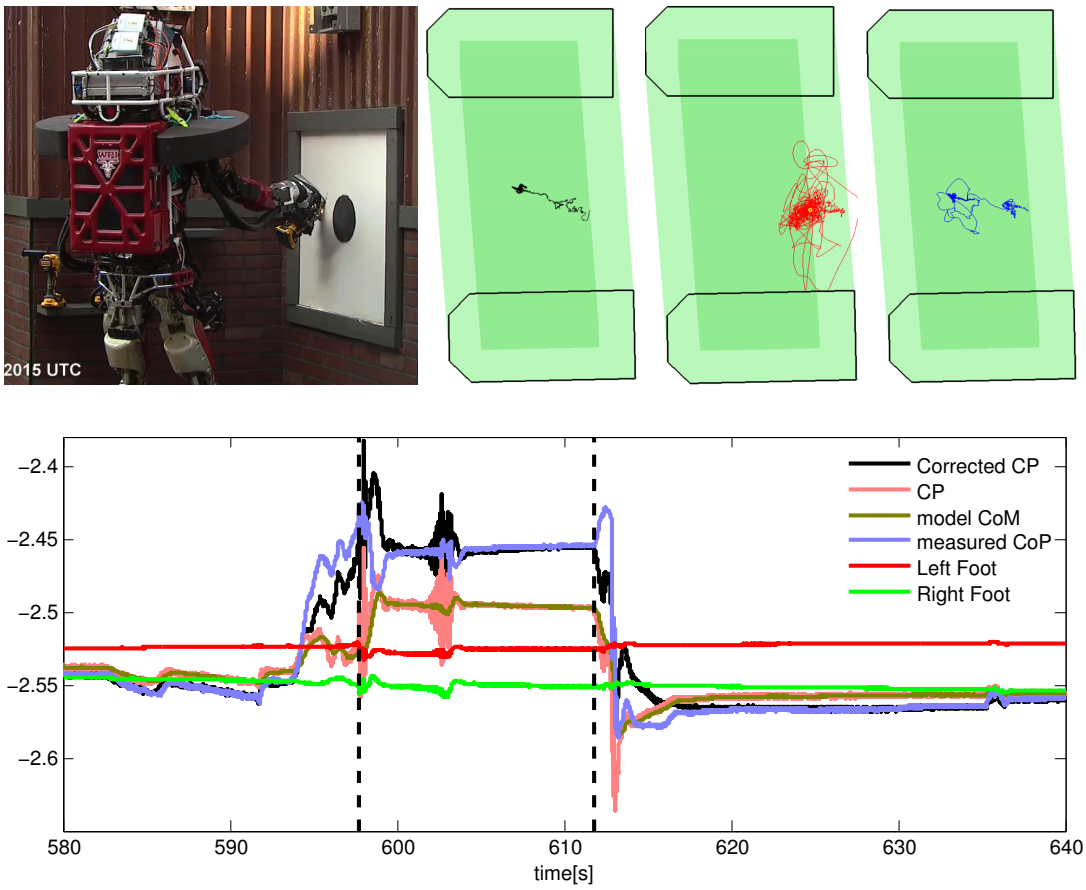

Figure 8: Top Left: Robot posture after error detection. Top Right: Black trace: The corrected capture point (CCP) up to the error detection, Red trace: The CCP during the "frozen" period, and Blue trace: The CCP moves back to the center of the polygon of support during manual recovery. Bottom: Plots of candidate fall predictors in the fore/aft direction during this event. The black vertical dashed lines mark the freeze time and the start of manual recovery.

fall warning too late, because the CoM velocity is not included. The capture point (CP) does not include information about external forces. The center of pressure (CoP) is noisy and gives too many false alarms. It can warn of foot tipping, but it is less reliable about warning about robot falling, which is not the same thing as foot tipping in a force controlled robot or if there are non-foot contacts and external forces. In this plot, we see that the CoP moves away from the safe region during recovery, predicting that the robot is falling again, while the corrected capture point (CCP) moves towards the interior of the safe region.

2.6 Motion Planning for Manipulation

In order to reliably complete manipulation tasks with a humanoid robot it is important to generate end-effector trajectories that maintain several constraints (such as Center of Mass (CoM) over the polygon of support and collision-free motions) simultaneously. One approach is to use a sample-based search algorithm, such as rapidly exploring random trees (RRT) (LaValle, 2006; Kuffner and LaValle, 2000; Karaman and Frazzoli, 2011), which can efficiently find a feasible path in the search space. However, searching in high DoF configuration spaces and performing post-processing such as smoothing are computationally intensive and time consuming.

Another category is optimization-based algorithms, such as CHOMP (Ratliff et al., 2009), STOMP (Kalakrishnan et al., 2011) and TrajOpt (Schulman et al., 2014), which can generate a path from an initial trajectory that may be in-collision or dynamically infeasible. The result of the trajectory optimization problem can be obtained in a short period of time, even for a high dimensional problem. Challenges for trajectory optimization are its sensitivity to the choice of initial guesses and getting stuck in local optima.

Since computation time was a high priority consideration in the DRC, and TrajOpt has the best speed performance in our benchmark tests compared with other optimization-base algorithms, we decided to adopt a modified TrajOpt as our motion planning approach and set up costs and constraints for each manipulation task to be performed by our Atlas robot.

For the manipulation tasks, the desired Cartesian motions for the robot are specified, such as, foot positions, hand positions and CoM locations. The motion planning optimizer formulates these motions as its costs and constraints, and computes a trajectory represented by the joint states at a set of waypoints. The general formulation of the optimizer is given by:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \geq 0, i = 1, 2, ..., n_{ieq} \\ & h_j(\mathbf{x}) = 0, j = 1, 2, ..., n_{eq} \end{aligned} \tag{3}$$

where f , g_i and h_j are scalar functions, n_{ieq} and n_{eq} are the number of the inequality constraints and equality constraints.

In our approach, we consider the robot kinematics only and represent the trajectory as a sequence of T waypoints. The variable $\mathbf{x}$ in (3) is of the form $\mathbf{x} = \mathbf{q}_{1:T}$, where $\mathbf{q}_t \in \Re^K$ describes the joint configuration at the t-th time step for a system with K DoF. The cost function $f(x)$ can be written as:

$$\begin{aligned} f(\mathbf{q}_{1:T}) = & \sum_{t=1}^{T} ((\mathbf{q}_{t+1} - \mathbf{q}_t)^T \mathbf{Q}_1 (\mathbf{q}_{t+1} - \mathbf{q}_t) + \\ & (\mathbf{q}_t - \mathbf{q}_{nom})^T \mathbf{Q}_2 (\mathbf{q}_t - \mathbf{q}_{nom}) + \mathbf{d}_\Delta^T \mathbf{Q}_3 \mathbf{d}_\Delta) \end{aligned} \tag{4}$$

where $\mathbf{Q}_1$, $\mathbf{Q}_2$, and $\mathbf{Q}_3$ are positive semidefinite weight matrices, and $\mathbf{q}_{nom}$ represents a nominal posture. These quadratic cost terms represent penalization of the weighted sum on the joint displacements between the waypoints, posture deviation from a nominal posture in joint space and posture error in Cartesian space. The first term can limit the movement of the robot and smooth the trajectory. The second term is used

to satisfy desired joint variables when all the constraints have been met. Similarly, the third term is used to push links to specific positions and orientations. The posture error in Cartesian space can be obtained by calculating the distance $\mathbf{d}_\Delta$ from a given configuration to a task space region introduced in (Berenson, 2011).

The constraints for the optimization problem include:

- Joint limits constraint. These can be written as $q - Q^- > 0$, and $Q^+ - q > 0$, where q is the set of reachable joint values, Q^+ is the maximum value in Q, and Q^- is the minimum value.
- Joint posture constraint, which is represented as $q - q_d = 0$. This constraint can lock the joint in some specific value q_d at some time steps.
- Cartesian posture constraint. This constraint can be established by setting the posture error $diag(c_1, c_2, ..., c_6)\mathbf{d}_\Delta = \mathbf{0}$, where $diag(c_1, c_2, ..., c_6)$ is a 6x6 diagonal matrix with diagonal entries from c_1 to c_6. We can relax some position and orientation constraints through setting the related entry to 0.
- CoM constraint. The horizontal projection of the CoM is desired to be on the support convex polygon between the two feet.
- Collision avoidance constraint. Generating a collision free trajectory is one of the most important features of a motion planner. But it is difficult to formulate collision constraints in a closed form for optimization-based motion planning. We use a hinge-loss function introduced in (Schulman et al., 2014) to set up the collision constraints based on the calculation of the signed distance for overlapping pairs of objects. The advantage of the method is that it can not only check for discrete collisions on each step but also integrate continuous-time collision avoidance constraints.

Therefore, to generate feasible motion for our Atlas robot, a variety of costs and constraints are set, such as costs for joint displacement, knee and back angles, pelvis height and orientation, torso orientation, shoulder torque, and constraints for joint limits, self-collision, environment collision avoidance, feet position and orientation, and CoM location. These are general costs and constraints. Task specific costs and constraints are also used.

To illustrate our approach, we briefly discuss the Door Task which was critical to complete in order to be able to advance to the manipulation tasks inside the building (Banerjee et al. 2015a). For push door, two trajectories need to be planned, which are approaching the door handle and turning the door handle. For a pull door, among two previous motions, the robot also has to pull the door back and block the door with the other hand.

1. A Cartesian posture constraint is added on the final step of the trajectory for approaching the door handle. The parameters of the constraint are the desired position and orientation for the robot end effector to grasp the handle, which are computed based on the handle configuration detected by the robot vision system.

2. During the handle turning motion, the handle hinge is not translating. There is only the rotation movement on the hinge. Two Cartesian posture constraints are added on the trajectory. First is the final step constraint. The offset transform T_2^1 is from grasper frame C_1 to the current handle hinge frame C_2, while the target frame C_3 is the current handle hinge frame rotating around 80∘ (see Fig. 9). The second one is a whole trajectory posture constraint, which limits the movement of the current handle hinge frame and only allows it to move along the hinge axis through setting the coefficients of the posture constraints $diag(c_1, c_2, ..., c_6)$ as $diag(1, 1, 1, 1, 0, 1)$. When the handle is held in the grasper, the transform T_2^1 can be obtained by adding a minor offset to the current gripper frame configuration calculating by forward kinematics.

3. Similar to the handle turning motion, opening the door also has a rotation-only point which is the door hinge. Hence, the offset transform T_2^1 can be calculated using the width of the door, and

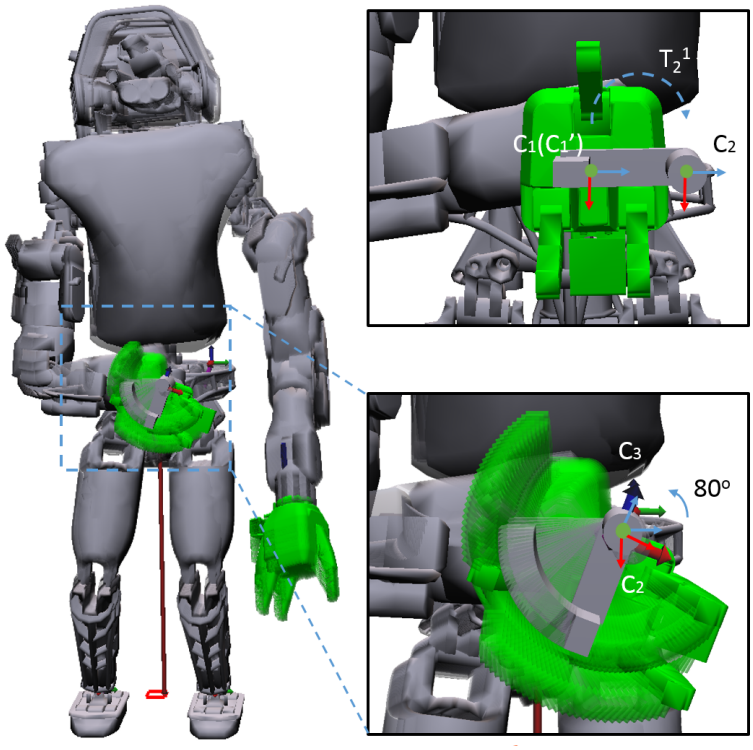

Figure 9: Visualization of constraints to generate the door handle turning motion

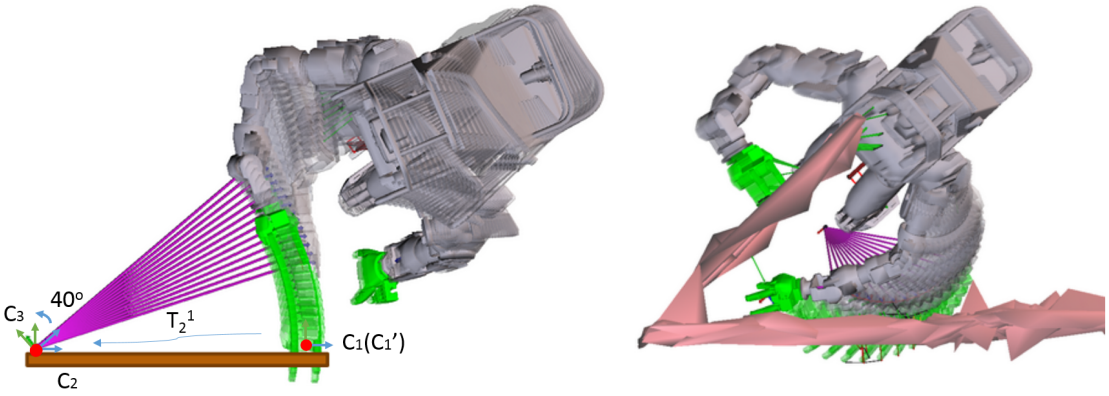

Figure 10: Constraints on generating door pulling motion (Left); hand inserting trajectory (right)

the target frame C_3 is the current door hinge frame C_2 rotating around 40° (see Fig. 10a). The movement of the current door hinge frame needs to be constrained to be only able to rotate along hinge axis.

4. After the robot pulls the door out, it needs to insert the other hand to prevent the door from closing again (see Fig. 10b). There are two Cartesian posture constraints that have to be added. The one is fixing the hand grasping the handle during the whole motion and the other one is moving the other hand to a target position behind the door which can be defined according to the position of the door handle.

Since the optimization problem we formulated involves collision-free constraints which are highly non-convex, the solver may get stuck in infeasible local optima. In our motion planning optimizer, the default method to generate an initial guess is linear interpolation. However, the solver may not be able to obtain a solution from the default initial guess. For example, after pulling out the door using the right hand, the robot needs to insert its left hand to block the door so it can not close again. The environment geometry is generated by convex decomposition (Mamou and Ghorbel, 2009) of point clouds (see Fig. 10b). The initial guess trajectory shows that the left hand has a collision with the wall. In this case, the solver can not find a feasible solution. We use multiple random initializations to help the algorithms escape from the case of local optima. The random initializations can be generated by inserting a couple of random states between the current state and the final desired state and connecting all these states through linear interpolation. Applying this method, the optimizer can find the solution successfully when it is used to generate motions in all DRC scenarios. A disadvantage of using multiple initial guesses is time consuming. To deal with it, we save the computed feasible trajectory and use it as initial guess next time.

Once the optimization problem is solved, the computed trajectory is sent to our low-level full body controller (Feng et al. 2015a). The controller traces the trajectory and computes physical quantities for each individual joint such as joint position, velocity, acceleration, and torque. Some of these outputs are then used as references in the joint level servos on the robot. We will now discuss some specific lessons learned about behavior planning.

Inverse kinematics is still a difficult problem. Tuning optimization-based inverse kinematics algorithms to avoid wild motions of the arm and still achieve task goals proved difficult, and constraint-based inverse kinematics algorithms eliminated too much of the workspace. We need more work in this area to achieve human levels of performance in trading off desired end effector motion in task space and undesired motion of the rest of the arm.

More degrees of freedom makes motion planning much easier ($7 \gg 6$). The original six degree of freedom arms on Atlas led to many problems finding reasonable inverse kinematic solutions. Adding an additional arm degree of freedom and including the whole body in manipulation planning greatly reduced the challenges of inverse kinematics. Similarly, a neck that allowed head turning (rotation about a vertical axis) would have been very useful. The Atlas robot only allowed head nodding (rotation around a horizontal axis in the frontal plane (a pitch axis)).

Approximate self-collision detection was adequate for walking. We used crude capsule-based approximations of the robot link shapes to avoid self-collisions during walking.

Planning contact-rich behaviors is an unsolved problem. It is startling to realize that we and all other teams failed to use the stair railings, put a hand on the wall to help cross the rough terrain, or grab the door frame to more safely get through the door in the DRC Finals. Even drunk people are smart enough to use nearby supports. Why didn't our robots do this? We avoid contacts and the resultant structural changes. More contacts make tasks mechanically easier, but algorithmically more complicated. Our contact-rich egress compared to other team's contact-avoiding strategies is a good example of this (Liu et al., 2015), as was our ladder climbing in the DRC Trials (DeDonato et al., 2015).

2.7 Perception Capabilities

Since "indoor" manipulation tasks suffered from communications blackouts, it was essential to develop perception capabilities for accomplishing the DRC Finals tasks. We supported sensor fusion and object/environment mapping. LIDAR information was combined with stereo vision. The user could identify targets by using the mouse to designate parts of displayed images and point clouds, and our perception code supported transforming this "scribble" into a useful object segmentation.

2.7.1 Odometry based Assembler

The Multisense SL on Atlas had a stereo camera pair and a spinning LIDAR as the primary sensors. The stereo camera pair generated disparity images using semi-global block matching. The spinning LIDAR generated range observation based on time of flight method. Both these observations can be converted to depth but they have different sensor noise and density. The LIDAR based depth measurements decreases in density radially from the spinning axis but have less sensor noise. On the other-hand the depth measurement from the stereo camera are dense within the viewing frustum but have relatively more noise. Since the LIDAR is a scanning sensor and the measurements are observed by movement in 6 dimensions and needed to be compensated in time to prevent motion distortions. Using the pose estimates provided by the kinematic state estimator (Xinjilefu et al., 2014) the LIDAR measurements are registered to a fixed world frame and time compensated. A shadow filter was used to remove noisy points that occur along the edge of the objects. Next a self filter was used to remove points from LIDAR scan that fall on the robot. It modeled the robot as sets of cylinders and used a ray collision check to determine if a point is within the cylinders that represent the robot. A moving average intensity filter was used to remove noisy laser observations due to specular reflection in a local neighborhood. Figure 11 shows the applications of the filters and the generated assembled and filtered LIDAR point cloud. The assembled and filtered LIDAR point cloud was combined with the disparity image from the stereo camera to generate the final fused depth measurements.

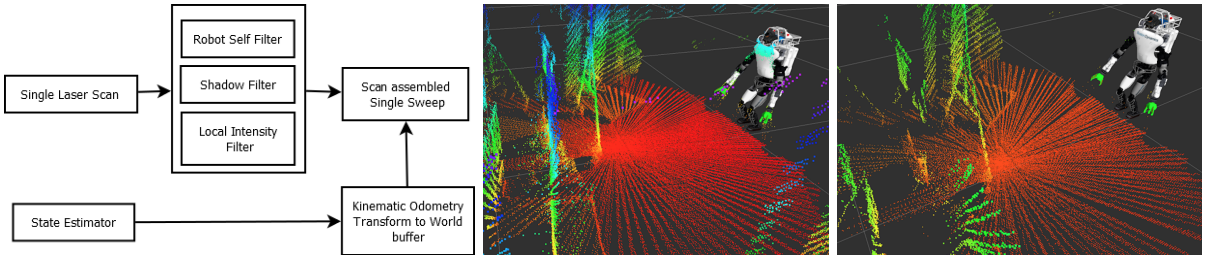

Figure 11: Left: Block diagram showing the generation of assembled point cloud. Right: Comparison of assembled point cloud without and with filtering. Notice the remove of spurious points from the robot body and the edges of objects in the scene.

2.7.2 Fused Depth Image

The LIDAR gives observations of the world whose density decreases radially as we move away from the LIDAR spinning axis. On the other hand, a stereo camera gives a dense observation of the world but has sparse measurements in areas with little texture. By combining a stereo camera and LIDAR data it is possible to get a more accurate depth observation of the world. An approach similar to a weighted joint bilateral filter (Matsuo et al., 2013) that reduces discontinuities in an RGBD image is used for smoothing and fusing the depth information. A non-linear joint bilateral filter (Kopf et al., 2007) is used to combine the laser and the stereo camera depth image. Each pixel depth is a weighted sum of the contributions from the neighboring pixels based on the spatial and photometric difference observed in the intensity image. This approach is based on the idea that two pixels have similar photometric value when they are close to each other spatially. This can be used to fuse the laser observation and the stereo disparity information to generate a dense depth map with fewer discontinuities.

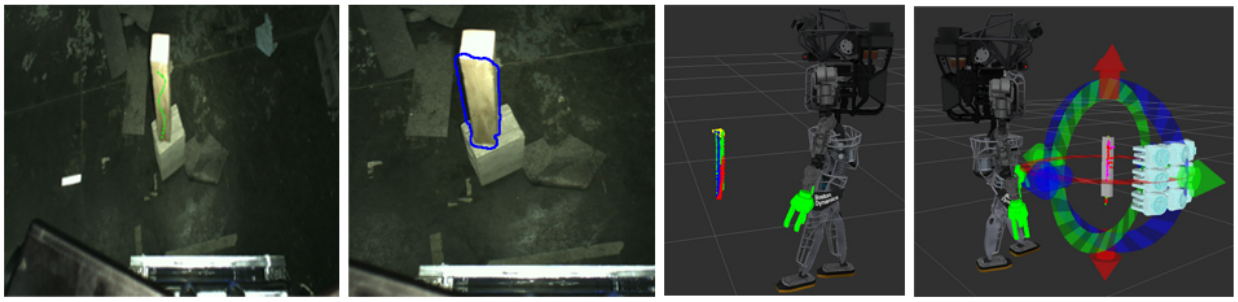

Figure 12: The scribble based object segmentation. From Left to Right, 1. The user marks the object (debris) using a scribble (green). 2. A Gaussian pdf based adaptive thresholding is performed. 3. Candidate point cloud is extracted. 4. Model fitting and grasp selection.

2.7.3 Object Detection

Similar to an approach used in interactive video segmentation (Bai and Sapiro, 2007), a scribble based object detector was used to segment and track objects from the scene. The seed points generated by the scribble are enriched with points from the local neighborhood. A mixture of Gaussian probability density functions is used to model the properties of the selected points from the reference image. The properties modelled are point color channels (L,A,B) and the fused depth. When a new image is received based on the mixture of gaussians probability density model, a weight map is generated. An adaptive threshold is applied using the weight map to generate the final segments. Additionally, The threshold is also weighted by the spatial distance from the reference scribble points. The segmented image points are converted into a fixed frame reference point cloud using the depth image. A model fitting is performed on this segmented cloud to generate the final aligned object segment (Figure 12).

2.7.4 Door detection

To achieve reliable door detection given the importance of the task (i.e., the task could not be skipped), two approaches for detection were developed in parallel; a fully automatic door detection algorithm, and a door detection algorithm that used operator scribble on the door handle as seed (Banerjee et al., 2015).

Autonomous door detection. The algorithm uses the geometrical features of the door to achieve door detection, i.e., it segments out parallel vertical lines, finds the perpendicular distance between them and then accepts those lines which have a distance equal to the door width within a reasonable threshold between them. The 2D image's contrast is increased and then filtered using a bilateral filter. Edges are detected using the Canny edge detector and Hough Transform is applied to detect lines of which only the vertical lines are segmented out. These lines are grouped into pairs to form probable door candidates. The entire image is reprojected into 3D. The distance between the probable door candidate lines are found and if they fall within the door width threshold, are kept. Door candidate line pairs having the same line equations are merged together. Final validation is performed by checking whether there exists a solid plane between the door lines (see Figure 13). For determining the handle, Connected Component Analysis (CCA) is used to filter out the region having the same door color, i.e., the handle is left out. Depending on the prior knowledge of the side where the door handle is, a search is performed from the bottom to locate regions inside the door candidate not filtered out by CCA (Figure 13).

Operator aided door detection. The operator scribbles over the latest 2D image from the Robot vision system, marking the position of the handle (see Figure 14) and based on the knowledge of where the door handle is, i.e. on the right or on the left of the door, points offset on the side of the seeded point is sampled and a plane is fit to appropriate those points. This provides the handle point (operator seed) and the normal to the door which coupled with the knowledge of the side where the handle gives the door estimate

Figure 13: Detected door and handle in a complex scene.

(Figure14).

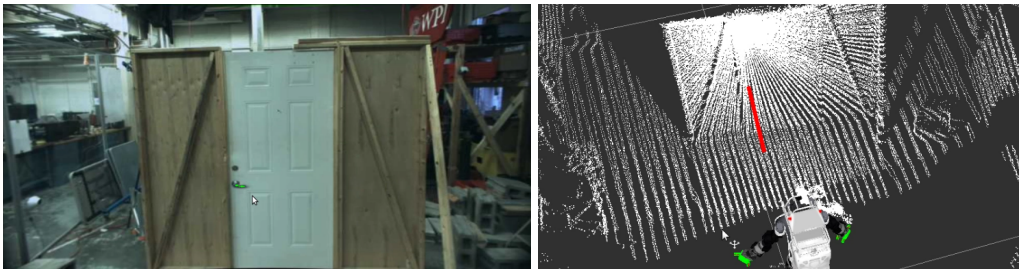

Figure 14: Operator seed in green color on the handle of the door (near cursor) and the detected door normal.

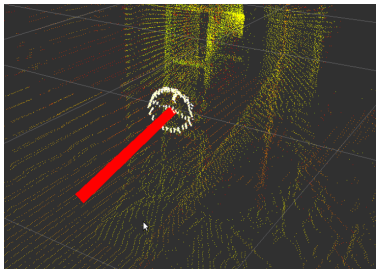

Figure 15: Detected valve normal shown in red.

2.7.5 Valve detection

The valve detection algorithm was based on fitting circles to the valve projection on a 2D plane. The operator seeded the center of the valve by scribbling in the latest available image. Scribbling in this context involved positioning the mouse at the center of the valve, and while holding the left button moving a few pixels about the center, with the seeded point from the scribble being the centroid of the scribbled points in 2D. This point was reprojected into 3D to get an estimate as to where the center of the valve would be. A nearest neighbor search was then performed on the center point in the 3D point cloud to find out the neighboring points. Since the nearest neighbor captured points behind the valve, i.e. points corresponding to the shaft of the valve and the mounting surface, a dot product of the vector from the center of the valve to the viewpoint with the vector from a neighbor to the center of the valve gave an estimate of the angle of deviation of the neighbors. A threshold based on experimentation was used to eliminate the points behind the valve. Then the valve normal was found out by performing a RANSAC fitting of a plane to the points that were left. Then all the points were projected on to this valve plane and a 2D circle fitting was performed which proved to be faster than fitting 3D circles. As a result, a more accurate center point of the valve was obtained and the algorithm output the center of the valve in 3D and the normal to the valve (Figure 15).

Figure 16: We established a DRC testbed in Worcester, MA for implementing our validation strategy in the last 45 days of the challenge.

Overall, our perception capabilities enabled the developers to implement perception based autonomous behaviors for various tasks and provided operators with enhanced visual feedback.

2.8 Validation Strategy

In preparation for the DRC Finals, we adopted a model-based design (MBD) approach in our software development for task validation. MBD is a powerful design technique that emphasizes mathematical modeling to design, analyze, verify and validate complex dynamic systems in which physical processes and computation are tightly integrated. In completing disaster relevant manipulation and mobility tasks for the DRC, we used an iterative MBD approach to specify and verify requirements for each task, develop models for physical human and robot actions as well as the environment, select and compose models of computation, simulate the human-supervised system as a whole, and verify and validate the algorithm designs on the physical robot. Our validation strategy can be described by a set of key features.

1. *Task factorization to move from teleoperation to supervised autonomy.* We evaluated each task, to the extent we could from the DRC rules specifications, to identify and implement methods to automate for faster and reliable completion. By testing and failing early, we were able improve the reliability and speed of each task. As an example, we improved the door task completion time from 45 minutes to under 8 minutes in the last month of the challenge.

2. *Implementation of a DRC Testbed.* Figure 16 depicts the DRC Testbed we built at the top floor of a parking garage in Worcester, MA. Its design was inspired by the DARPA testbed event that took place in South Carolina in March 2015. Through modular and reconfigurable designs, we were able to test each task and combinations of tasks in various configurations. Team WPI-CMU operators and developers refined their skills and methods for completing the DRC tasks in the last 45 days of the challenge an the testbed.

3. *Models of Computation.* We designed and implemented event-driven finite state machines (FSM) for most tasks that included critical subtasks. For example, we split the Door Task into four sub-tasks; door detection (DoorDetect), approaching the door (Approach), door opening (Open), and walking through the door (GoThrough). To maximize the utility of the human-robot team to complete this task, we iteratively designed a factoring of the task between the human operator and robot. This factoring leverages the superior perception and decision making capabilities of the human operator to oversee the feasibility of the steps planned in walking, selecting one of the three door types, and enabling the operator to make adjustments to robot actions to minimize errors. In the meantime, robot intelligence handles the balancing, motion planning and detection algorithms. With the subtasks as the states of the FSM, this framework allowed our team to control the autonomous execution of the process with human supervision and validation at critical steps. Figure 17 shows the FSM design with human input as an event resulting in state transitions.

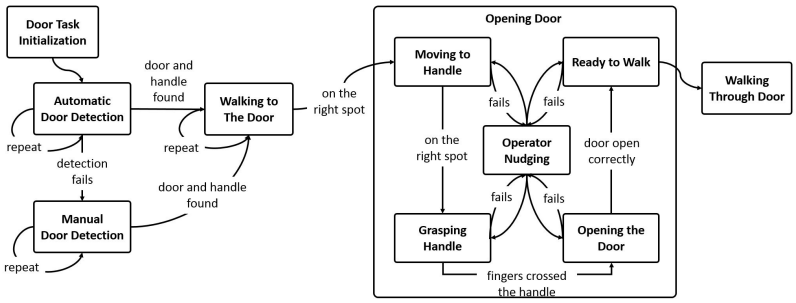

Figure 17: An event-driven finite state machine is designed for most tasks. The figure depicts the FSM design for the Door task.

Figure 18: Driving and egress (getting out of the car) tasks.

3 Approaches To Specific Tasks

3.1 Driving

Figure 18 shows our robot driving the Polaris X900 vehicle in the DRC Finals. We won the "best driver" award in the DRC Trials (DeDonato et al., 2015). For the Finals, we developed a new mechanical tool for the robot to use to turn the steering wheel, and procedures to compensate for kinematic modeling errors so that the steering wheel turned smoothly. We provided the operator with a plan view for driving based on visual odometry, LIDAR, and IMU information. There are two components to driving: speed control and steering. Remote operators could easily steer the vehicle even with considerably delayed feedback. However, the operators found it difficult to control speed by commanding the robot to press the accelerator, so we set up an autonomous speed control behavior. Stereo camera information was used to generate an estimate of the velocity of the vehicle. In visually dense environments, the stereo cameras with a modified version of the libviso2 package (Geiger, 2012) provided very good data. The desired velocity was provided by the operator and was passed to a PI controller that actuates the throttle using the ankle joint of the robot. Once set, the robot drives the vehicle at a steady and slow pace autonomously. The improved display and the autonomous speed control behavior is what we believe made driving reliable. Driving worked reliably on both days of the DRC and in many practices before the DRC. Further details can be found in (Knoedler et al., 2015).

3.2 Egress

Our strategy for getting out of the car was to maximize contact with the car, as well as provide some mechanical aids (Figure 18). Maximizing contact led to using the car to stabilize the robot as much as

possible. It is interesting that other Atlas teams tried to minimize contact with the car, spending a lot of time carefully balancing on one foot with no other supporting or stabilizing contacts. MIT released a video of their robot standing on one foot while the vehicle is being shaken (MIT, 2015b). These are impressive but perhaps unnecessary demonstrations of high quality robot control. Both IHMC and MIT added a handle to the car within easy reach of the robot to hold while driving, but their robots did not use it to get out of the car.

In terms of mechanical aids, we used wooden blocks to make sure the robot sat in the seat the same way each time, as the bottom of the pelvis was rigid, small, and had little friction with the hard seat, so the robot easily slid and rotated. We provided an accelerator pedal extension so the robot could drive from the passenger side, as it could not fit behind the steering wheel. We added a wooden bar to replace the missing car door that could have been used as a support. We provided a step since the robot's leg could not reach the ground when seated or standing in the car.

We manually decomposed the task into phases with different contact sets, desired poses, and desired trajectories. We optimized the sequence of static robot poses as well as contact locations that satisfied constraints such as geometric, joint limit, and equilibrium constraints. To accommodate modeling errors, uncertainties, and satisfy continuous contact constraints, we used online robot pose optimization to generate smooth trajectories based on sensor feedback and user inputs. Our motion planning method generated a rough plan for the challenging car egress task of the DRC. Combined with online robot pose optimization, the rough plan could be applied to real-time control with excellent performance. Our egress worked reliably four out of four times at the DRC Finals (rehearsal, battery testing, day 1, and day 2) as well as during many practice tests before the DRC. Further details are presented in (Liu et al., 2015).

3.3 Door

Door traversal can be broken down into four sub-tasks; door detection, walk to the door, door opening, and walk through the door. Door detection locates the door and finds its normal. Wrist mounted cameras assisted in finding the door handle, maintaining contact, and providing information to the human supervisor. The handle is turned, and in the DRC Finals with a push door, the other arm is used to push the door open. We had the robot walk sideways through the doorway to maximize clearance, taking into account the large sideways sway of the robot as it walked (Figure 6). This behavior worked four out of five times at the DRC Finals including the battery test and the rehearsal. The one failure during the rehearsal was safely handled as described previously. More details are available in (Banerjee et al., 2015).

3.4 Valve

The valve task involved approaching and turning a valve (Figure 19). The robot was manually commanded to go to the vicinity of the valve. The operator marked the valve with a line using a mouse in one of the camera images. The valve was then automatically located and segmented in the image. We took the approach of inserting fingers into the interior of the valve handle, while some teams grasped the handle rim. This task was performed twice at the DRC finals, and is one of our most reliable tasks.

3.5 Drill/Cutting

The drill/cutting task re-used many components of the valve task. The robot was manually commanded to go to the task area. The operator marked a drill with a line using a mouse in one of the camera images. The drill was then automatically located. We developed a two handed strategy to grasp the drill and rotate it so the switch was exposed. We implemented force control based on the wrist force/torque sensor to ensure a reasonable contact force to fully insert the cutting drill bit but not push the robot over.

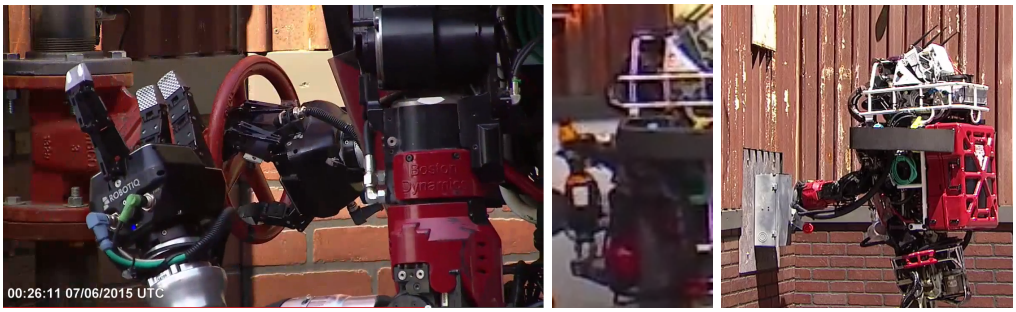

Figure 19: Rotating the valve, reorienting the drill and regrasping it to turn it on, and having completed the "push the switch down" surprise task..

Both MIT and our team used a two handed grasp and object reorientation strategy. This was a huge mistake for both teams. For us, although this strategy was more reliable than our one handed strategies if both electric forearms of the robot were working, it was rarely the case that both forearms were working. For example, on the morning of day 2 of the Finals we replaced the left forearm because it had failed, and the right forearm failed during the drill task. IHMC, MIT, and WPI-CMU all had to replace faulty forearms during the DRC Finals. We had little practice time in the months we had the new forearms before the DRC as one or both were often not working. We should have developed a one handed strategy that used the shelves as another hand to hold the drill during regrasping (as some teams did), or at least have a one handed backup strategy in case of forearm failure. We do not have statistics on how well we can do this task, as we were rarely able to test it due to forearm unreliability. We did the other manipulation tasks (door, valve, surprise) one handed, and were able to do them with either hand. For MIT, on day 1 a fall broke one of their forearms, and they had to skip the drill task that day. IHMC used a one-handed strategy and succeeded at the drill task both days at the DRC Final.

3.6 Terrain and Stairs

We have already described our approach to controlling robot walking. It was useful for the terrain and stairs task to take into account the change in height of the robot. To decrease the risk of falling we limited the step size of steps going down, as these steps are the most difficult for the robot, due to ankle range limits and knee torque limits. The calf hitting the next tread is a problem for Atlas robots walking up stairs. We used a splayed foot strategy to reduce this problem, while other Atlas teams walked on their toes. We also used high oil pressure and a task specific planner for the stairs task. Figure 20 shows the terrain and stairs tasks, as well as stepping off the platform during car egress. Each of these tasks, including stepping off the egress platform, was performed twice at the DRC Finals. The stair task was only fully executed at the DRC Finals, as our safety belay system was not tall enough to allow full stair climbing and we did not have enough faith in our software or robot to do it without a belay before the DRC Finals. This lack of trust in our own robot performance does not bode well for deployment of humanoid robots in the near future, unless there is a paradigm shift in both hardware and software design. More details on walking are available from (Feng et al., 2015). More details on a paradigm shift are available from (Atkeson, 2015).

3.7 Debris

Although we did not perform this task in the DRC, we were prepared to do the debris task with a shuffling gait with small vertical foot clearance. In tests, this approach sometimes failed because the pieces of debris jam against the walls and each other. We felt the terrain task was easier and had a higher expected chance of success.

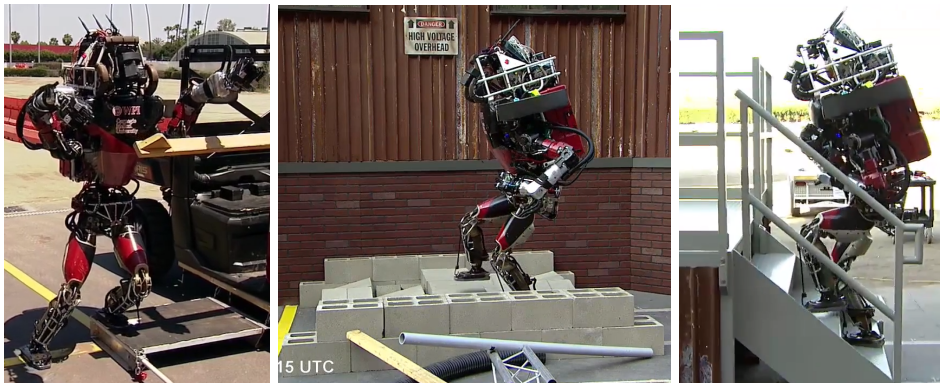

Figure 20: Stepping off platform, Walking over terrain, Climbing stairs

4 How we achieved reliability

We focused on competing with the other Atlas robots in the DRC, as we were limited in how much we could modify or improve our robot mechanically to optimize performance for the DRC tasks. We believed falls would be fatal to the Atlas robots, so we chose a strategy that minimized fall risk, and also physical human intervention (resets). We believed other Atlas robot teams would attempt to go too fast leading to operator and robot errors, falls, and damage to their robots. Our strategy to avoid falls and resets included

- Extensive operator practice and concurrent system testing.
- An explicit "slow and steady" strategy that emphasized getting points rather than minimizing time. Our plan was to go slowly enough to reduce the risk of falling to an acceptable level and complete all tasks within the allotted time (1 hour), In fact, our first run was completed with about 30 seconds left in the hour.
- Explicit monitoring for robot errors and anomalies, and safety code to safely stop the robot if a fall was anticipated.
- Adding additional superhuman sensing in the form of hand cameras on the wrists and foot cameras on the knees looking down. This is a first step towards our planned "whole-body vision system".
- Enabling the operator to control and monitor the robot at varying degrees of abstraction.

Our prediction that other Atlas teams would rush and fall was correct. Both the MIT and IHMC Atlas teams suffered from operator errors on the first day (MIT, 2015a; IHMC, 2015). MIT performed worse on the second day, due to damage from the first day and difficulty in repairing the robot, as expected (the manufacturer of the robots, Boston Dynamics, performed all repairs). IHMC was able to get the damage from its two falls on the first day repaired, and avoided operator errors on the second day, resulting in an impressive 2nd place performance on day 2. Other Atlas teams in the DRC Finals did not do as well.

We successfully avoided falling and remote operators could safely recover from difficult situations. We were the only team in the DRC Finals that attempted all tasks, scored points (14/16), did not require physical human intervention (a reset), and did not fall in the two missions during the two days of tests. We also had the most consistent pair of runs.

We could have done more in terms of good systems and software engineering process. Time constraints led us to minimize planned systems and software engineering development and quality assurance processes, including the use of tool suites, requirements analysis, analysis of alternatives and trade-offs,

and automated test generation and verification. Instead, we focused on getting one primary approach implemented, working, and tested. We used manual and ad-hoc test designs and procedures. Our plan if we do this again is to use better systems and software engineering tools and processes.

We could have done more to provide early fall predictions and detect faults and anomalies. In manipulation, we tracked the corrected capture point. In walking, the corrected capture point moves around quite a bit, and is less reliable as a fall predictor. Other signals can be fused to improve fall prediction. Vertical foot forces (F_z) too high or not high enough, and other foot forces and torques being large are warning signs of large external forces, Joint torque sensors can be used to attempt to locate a single external force to either a hand (manipulation), a foot (tripping), or another part of the body (collision). Robot skin-based sensing would have been and can be expected to be extremely useful as part of an early warning system. Horizontal foot forces (F_x, F_y), yaw torque (torque around a vertical axis: M_z) going to zero, foot motion due to deflection of the sole or small slips measured using optical sensors, and vibration measured in force/torque sensors or IMUs in the foot are early warning signs of possible slipping. The center of pressure going to a foot edge and foot tipping measured by a foot IMU are early warning signs of individual foot tipping and resultant slipping or possible collapse of support due to ankle torque saturation. Any part of the body such as the foot or hand having a large tracking error is a useful trigger for freezing the robot and operator intervention.

We can do better in terms of control. One of our goals is coming closer to human behavior in terms of speed, robustness, full body locomotion, and versatility. A very simple step recovery behavior based on the corrected capture point has been implemented. We think high cadence dynamic walking, heel strike, and toe push off are essential to fast robust walking. To achieve this, better system identification will further bridge the gap between simulation and hardware and improve the overall performance. Optimizing angular momentum in the high-level controller will also benefit fast walking and will increase the safety margin for balancing. We also need to design reflexes for handling slips and trips.

Hardware reliability is a limiting factor for humanoids. Humanoid systems consist of many components, any of which can fail. As a result, our ATLAS robot had a mean time between failures of hours or, at most, days. Since we could not repair the robot ourselves, we often ran the robot with various components not working. We had to develop behaviors that tolerated or worked around hardware failure. In particular, it was rare that both electrical forearms in the final Atlas configuration were both working at the same time. As previously discussed, we should have done a better job developing behaviors that worked when unreliable components actually turned out to be not working, rather than engage in wishful thinking that the problems would get fixed or go away before the DRC.

Getting up after a fall. Standing up after a fall is very difficult and requires substantial torso and upper body strength. If the fall is on anything other than a flat level surface, such as rough terrain, stairs, debris, completely or partly in the vehicle, or against a wall, it is difficult to plan in advance. The current Atlas robot is too top heavy and its arms are too weak (similar to a Tyrannosaurus Rex) for it to reliably get up from a fall.

Design for robustness and fall recovery. General system robustness, robustness to falls (avoiding damage), and fall recovery (getting back up) need to be designed in from the start, not retro-fitted to a completed humanoid design. We believe lighter and structurally flexible robots with substantial soft tissue (similar to humans) are a better humanoid design. We have been exploring inflatable robot designs as one approach to fall-tolerant robots (Atkeson, 2015). We note that in the DRC rehearsal both IHMC and our team did not practice most tasks, since safety belays were not allowed. We did not have enough faith in our software or robot, and we believed a fall would be fatally damaging to our Atlas robots. In the DRC Finals it turned out (at least for IHMC) that fall damage was more easily repaired that we anticipated, but MIT was less fortunate. As mentioned previously, this lack of faith in our own robot performance does not bode well for deployment of humanoid robots in the near future, unless there is a paradigm shift in both hardware and software design (Atkeson, 2015).

Event	Status	Day 1	Day 2
Start		00:00	00:00
Driving	Completed	03:30	02:40
Egress	Completed	11:50	09:19
Door Task	Completed	20:00	17:43
Valve Task	Completed	28:10	22:08
Wall Task	Aborted	36:50*	30:56**
Surprise Task	Completed	44:00†	39:57‡
Terrain Task	Completed	56:40	46:59
Stairs Task	Completed	59:35	56:06

* Software bug
** Forearm failure
† Switch Task
‡ Plug Task

Table 1: Completion times for two days of runs at the DRC Finals.

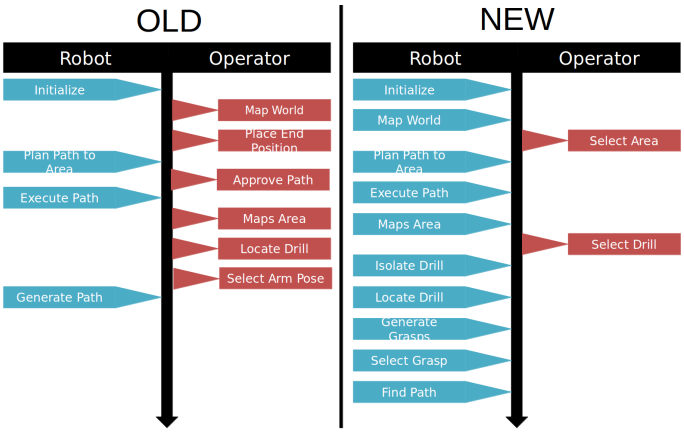

Figure 21: Robot/operator work balance: Left: DRC Trials; Right: DRC Finals.

5 Overall Results

We were the only team in the DRC Finals that tried all tasks, did not require physical human intervention (a reset), and did not fall in the two missions during the two days of tests. We also had the most consistent pair of runs. Our task performance was not perfect. We scored 14 out of 16 possible points over the two runs at the DRC Finals. We had a program design error and an arm hardware failure that caused two attempts at the Wall task to fail. However, unlike other teams, these failures did not cause our robot to fall. The operator simply was able to move on to the next task. Table 1 summarizes our team's runs at the DRC Finals.

Figure 21 shows the increased level of autonomy in the DRC Finals in 2015 compared to the DRC Trials in 2013 in the drill task.

6 Lessons Learned

Through our involvement in the DRC for more than three years, we have learned valuable lessons on theory and practice of humanoid robots for disaster response. We report the following as a summary of our findings.

Reliable hardware is critical. When our team received the Atlas robot after our performance in the Virtual Robotics Challenge in June 2013, we made a prediction that the robot would be functional and available for testing and development about 50% of the time. Even though we did not collect hard data to verify this assumption, we estimate that our robot hardware was functional for about 50% of the total time or maybe slightly more. Down times were caused by various reasons including upgrades, failures caused by degradation of hardware and failures caused by operator errors. In order to provide practical robot systems for real disaster response scenarios, it is essential to have reliable robot hardware. This includes powerful and robust universal grippers that can perform manipulation tasks. One way to mitigate hardware failures in real disaster situations would be to design with redundancy both in terms of hardware and in terms of behaviors.

Reliable software is critical. On Day 1 at the DRC Finals, when our robot dropped the drill after turning it on, we tracked the problem back to the first time occurrence of a software bug. Team WPI-CMU's journey in the DRC mostly involved designing software capabilities for the Atlas robots. Even though we strived for best practices in software development, the number of developers with varied backgrounds has become a challenge for our team. It may not be possible to achieve perfection but a practical disaster response robot needs to be programmed to have persistent behaviors to complete the challenging tasks. It is also critical to equip robots with safety software as we discussed earlier. Testing early and often are also important in reliable software development.

Factoring tasks to maximize the utility of the human-robot team is essential. It is critical to pursue more autonomy in the development of robots for disaster response as set forth by the overarching goal of the DRC. The real disaster scenes will be much more unstructured and chaotic than the simulated environments we experienced at the finals. However, the current state-of-the-art as demonstrated by the participating teams only allows for short term and partial autonomy. In our approach, the key to our performance in the DRC Finals was factoring of the tasks between the human operator and the robot. We iteratively improved reliability and speed going from teleoperation to supervised autonomy by providing capabilities to the human operator to intervene with robot decisions at the key steps. For example, the operators were able to approve the steps created by the step planner, accept or reject the trajectories generated by the motion planner, or in some cases nudge the end-effector pose for improved manipulation. Our models of computation enabled this general approach to maximize the utility of the team.

A rigorous validation strategy and operator training is essential. Anecdotal evidence suggests that this is a lesson learned for all the DRC Finals participants. Executing a rigorous validation plan improved our team's performance. We established a full course DRC testbed and ran tests outdoors in the last 45 days of the challenge. We were able to identify opportunities to speed up the task completion. Furthermore, our operators had extended periods of time in training and they built confidence in their skills as well as the robot. Our software architecture and computational models enabled the execution of our validation strategy both in simulation (early on and when the robot is down) as well as on the real robot. We were able to unify the tasks leading to full course testing prior to finals and we were able to enhance the reliability. In critical tasks such as the Door Task and Vehicle Egress, we improved our robot's reliability (percentage of successful task completion) and speed (time of task completion) at least by a factor of 10 through our validation strategy.

Early detection of robot or operator errors can prevent falls. Often falling is caused by errors in robot locomotion or manipulation, where the robot pushes or pulls itself over, rather than outside perturbations such as wind, support movement, or external agents pushing the robot. Our field has over-emphasized research on responses to outside perturbations such as "push recovery" over handling robot-generated errors. We found that simply stopping what the robot was doing gave the remote operator enough time to successfully intervene. It is an interesting research question as to how to distinguish between robot errors and external disturbances, or combinations of the two, especially with no full body skin tactile sensing. The correct responses to different robot errors and external disturbances are often quite different and conflicting.

Operators want to control the robot at many levels. Our robot operators wanted to be able to control the robot at many levels, and rapidly and conveniently switch between them: 1) command joint velocities or changes in positions, 2) command Cartesian velocities or changes in positions, 3) designate end effector targets such as desired grasps or footsteps, 4) provide task parameters such as speed, and 5) select tasks or task components to perform.

We can do much better in terms of human-robot interaction. Supervisory control of the DRC robots was only possible by operators who had extensively practiced for months, and even then errors were made. We couldn't touch our robot without two emergency stops and rubber gloves to prevent 480V electric shocks. We could safely poke our robot with a stick. Robots and humans aren't really working together until control of a robot can be learned in minutes and physical interaction isn't life threatening, let alone easy and fun (Atkeson, 2015).

7 Conclusion

We discussed the details of our systems and software in our approach to controlling humanoid robots for disaster response missions. We utilized human-supervised autonomous robot behaviors to complete the DRC Finals mission. Our approach and validation strategy was effective as the Team WPI-CMU ranked 7^{th} out of 23 teams in the DRC Finals. Team WPI-CMU contributed to the field of disaster robotics by not only introducing new theoretical and practical knowledge but also by training a new cadre of students with multidisciplinary skills who are aware of the challenges that come with designing and developing robots for humanitarian aid and disaster response.

Acknowledgements

This material is based upon work supported in part by the DARPA Robotics Challenge program under DRC Contract No. HR0011-14-C-0011.

References

Atkeson, C. G. (2015). Big Hero 6: Let's Build Baymax. `build-baymax.org`.

Bai, X. and Sapiro, G. (2007). A geodesic framework for fast interactive image and video segmentation and matting. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8.

Banerjee, N., Long, X., Du, R., Polido, F., Feng, S., Atkeson, C. G., Gennert, M., and Padir, T. (2015). Human-supervised control of the ATLAS humanoid robot for traversing doors. In *Humanoid Robots (Humanoids), 15th IEEE-RAS International Conference on*.

Berenson, D. (2011). *Constrained Manipulation Planning*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

DeDonato, M., Dimitrov, V., Du, R., Giovacchini, R., Knoedler, K., Long, X., Polido, F., Gennert, M. A., Padir, T., Feng, S., Moriguchi, H., Whitman, E., Xinjilefu, X., and Atkeson, C. G. (2015). Human-in-the-loop control of a humanoid robot for disaster response: A report from the DARPA Robotics Challenge Trials. *Journal of Field Robotics*, 32(2):275–292.

Feng, S., Xinjilefu, X., Atkeson, C. G., and Kim, J. (2015). Optimization based controller design and implementation for the Atlas robot in the DARPA Robotics Challenge Finals. In *Humanoid Robots (Humanoids), 15th IEEE-RAS International Conference on*.

Geiger, A. (2012). LIBVISO2: C++ Library for Visual Odometry 2. `www.cvlibs.net/software/libviso/`.

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

IHMC (2015). Personal communication.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. (2011). Stomp: Stochastic trajectory optimization for motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4569–4574.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894.

Knoedler, K., Dimitrov, V., Conn, D., Gennert, M. A., and Padir, T. (2015). Towards supervisory control of humanoid robots for driving vehicles during disaster response missions. In *IEEE International Conference on Technologies for Practical Robot Applications*.

Kopf, J., Cohen, M. F., Lischinski, D., and Uyttendaele, M. (2007). Joint bilateral upsampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):to appear.

Kuffner, J. and LaValle, S. (2000). RRT-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 2, pages 995–1001 vol.2.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, U.K. Available at http://planning.cs.uiuc.edu/.

Liu, C., Atkeson, C. G., Feng, S., and Xinjilefu, X. (2015). Full-body motion planning and control for the car egress task of the DARPA Robotics Challenge. In *Humanoid Robots (Humanoids), 15th IEEE-RAS International Conference on*.

Mamou, K. and Ghorbel, F. (2009). A simple and efficient approach for 3d mesh approximate convex decomposition. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 3501–3504.

Matsuo, T., Fukushima, N., and Ishibashi, Y. (2013). Weighted joint bilateral filter with slope depth compensation filter for depth map refinement. In *VISAPP (2)*, pages 300–309.

MIT (2015a). Personal communication.

MIT (2015b). Egress robustness tests. `https://www.youtube.com/watch?v=F5CBRmDQXTk`.

Pratt, G. and Manzo, J. (2013). The DARPA robotics challenge [competitions]. *Robotics & Automation Magazine, IEEE*, 20(2):10–12.

Pratt, J., Carff, J., Drakunov, S., and Goswami, A. (2006). Capture point: A step toward humanoid push recovery. In *Humanoid Robots (Humanoids), 6th IEEE-RAS International Conference on*, pages 200–207, Genoa, Italy.

Ratliff, N., Zucker, M., Bagnell, J., and Srinivasa, S. (2009). Chomp: Gradient optimization techniques for efficient motion planning. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 489–494.

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*.

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

Xinjilefu, X., Feng, S., and Atkeson, C. (2015). Center of mass estimator for humanoids and its application in modelling error compensation, fall detection and prevention. In *Humanoid Robots (Humanoids), 15th IEEE-RAS International Conference on*.

Xinjilefu, X., Feng, S., Huang, W., and Atkeson, C. G. (2014). Decoupled state estimation for humanoids using full-body dynamics. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 195–201. IEEE.