NO FALLS, NO RESETS: Reliable Humanoid Behavior in the DARPA Robotics Challenge
DRAFT version 14

C. G. Atkeson*, B. P. W. Babu†, N. Banerjee†, D. Berenson†, C. P. Bove†, X. Cui†, M. DeDonato†, R. Du†, S. Feng*, P. Franklin†, M. Gennert†, J. P. Graff†, P. He†, A. Jaeger†, J. Kim, K. Knoedler, L. Li†, C. Liu*, X. Long†, T. Padir†, F. Polido*†, G. G. Tighe†, X. Xinjilefu*

***Abstract*— We describe Team WPI-CMU's approach to the DARPA Robotics Challenge (DRC), focusing on our strategy to avoid failures that required physical human intervention. We implemented safety features in our controller to detect potential catastrophic failures, stop the current behavior, and allow remote intervention by a human supervisor. Our safety methods and operator interface worked: we avoided catastrophe 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 discuss lessons learned from the DRC.**

I. Introduction

Our goal in this paper is to be provocative. We believe that robots are useless for disaster recovery if they require humans to rescue the robots, rather than the other way around. We were surprised at how many robots required physical human intervention during the DARPA Robotics Challenge (DRC) Finals in June, 2015. We are disappointed that the coverage of the DRC has been dominated by videos of robots falling down such as this video put out by our sponsor (DARPA, 140,000 views in one month) [1], and this video put out by the flagship magazine of the professional society sponsoring the Humanoids conference and which aspires to be the professional society for robotics and humanoid robotics (1,400,000 views in one month) [2]. It would have been much better to include some evidence of success in these videos, so the public would have a more balanced perception of what happened at the DRC [3].

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.

Our task performance was not perfect. We scored 14 out of 16 possible points over the two day DRC. We had a program design error and an arm hardware failure that caused two attempts at the drill/cutting task to fail. However, unlike other teams, these failures did not cause our robot to fall. The robot simply moved on to the next task.

Our safety code and human-robot control interface prevented several falls. Other team's robots fell or got stuck due to operator errors, robot errors, and hardware failures. If this had been an actual disaster in which physical human intervention was risky, costly, inefficient, or not possible (such as recovering from a nuclear or toxic chemical disaster, a "dirty" bomb attack, or a dangerous epidemic such as the recent Ebola epidemic), of the teams that attempted all tasks the only teams that would have proven useful or perhaps even survived the two missions over the two days of the DRC would be the CMU CHIMP team and our team, Team WPI-CMU. However, none of the robots, including ours, could currently deal with the complexity and difficulty of an actual disaster site such as the Fukushima Daiichi disaster site.

We expected bipedal robots to have trouble in the DRC. We used an Atlas humanoid robot built by Boston Dynamics. Compared to wheeled and quadruped robots, bipedal humanoids usually have higher centers of mass and a smaller support region, which results in a smaller stability margin and a higher risk of hardware damage when falling. In a real-world scenario where humanoid robots have to interact with the environment without a safety system such as a belay, fall prevention and recovery behaviors are necessary. Avoiding getting stuck and getting un-stuck are also important. The DRC attempted to mimic a real-world scenario, in which robots had to perform disaster response related tasks involving physical contact with the environment. What we observed during the contest was that most humanoids fell down and no walking humanoid that fell got back up. We were also surprised that wheeled robots had so much trouble with balance, falling, and getting stuck in the DRC.

This paper focuses on the top three Atlas teams: IHMC, MIT, and WPI-CMU. Due to page limits, our references will be largely limited to our own papers which provide both more details than this overview paper and extensive references. Please see our web page for the related papers (including those also submitted to this conference) and videos [4]. The accompanying video shows our final run on day 2 of the DRC Finals, illustrating our strategy for various tasks [5].

II. Our Strategy

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. Our strategy was to go slowly enough to reduce the risk of falling to an acceptable level, complete all tasks within the allotted time (1 hour), and implement fall prediction safety code to safely stop the robot if a fall was anticipated. 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 "slow and steady" strategy almost worked. Both the MIT and IHMC Atlas teams suffered from operator errors on the first day [6], [7]. MIT performed worse on the second day, due to damage from the first day and difficulty in repairing the robot, as expected. IHMC was able to repair the damage from its two

*Robotics Institute, Carnegie Mellon University
†Robotics Engineering, Worcester Polytechnic Institute

falls on the first day, and avoided operator errors on the second day, resulting in an impressive 2nd place performance on day 2. The IHMC robot did fall after completing the course on day 2, due to what the American National Football League penalizes as "excessive celebration" [8]. Other Atlas teams in the DRC Finals did not do as well.

III. Fundamental Behaviors

Optimal Control for Manipulation and Walking: 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) [9], [10].

In most cases, 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 Boston Dynamics provided joint level controllers.

It is necessary to generate joint level kinematic targets for the robot, since inverse dynamics alone performs poorly when facing modeling errors. For the DRC Trials in 2013, we implemented a full-body controller using independent inverse kinematics and inverse dynamics [9]. An independent inverse kinematics controller was suitable for the quasi-static motions in the DRC Trials, and was preferred for easy implementation.

However, we were concerned about inconsistency between the inverse kinematics and inverse dynamics modules due to their different sets of constraints. For the DRC Finals, two versions of the full-body controller were implemented, one for walking and the other for manipulation.

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. These design choices were motivated by the observation that the legs on Atlas have much better performance than the arms. The leg joints have less sensing noise and Coulomb friction and better torque control. For the arm joints, we were unable to use higher velocity gains and failed to achieve acceptable joint tracking without a position control loop. Explicit inverse kinematics is also preferred to double integration of accelerations to calculate desired positions for stability and accuracy reasons.

We used both precalculated trajectories and TrajOpt [11] to plan trajectories in real-time. In retrospect, we should have precalculated all trajectories and devised ways to parametrically adjust them online given the sensed target location. TrajOpt was slow, often produced unusable trajectories, and its solutions had to be carefully checked by a human operator..

Walking: Our current approach is to plan footstep locations, and then the controllers attempt to place the feet on those footstep locations. Page limits prevent us from describing our foot placement algorithms here. Early versions are described in [12] and the versions for the DRC Finals are described in [10]. 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) to optimize for a nominal center of mass trajectory, 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.

We no longer use inverse kinematics to generate joint level kinematic targets for walking. For the lower body, we numerically integrate the joint accelerations from the inverse dynamics into desired velocities, which are tracked by the joint level controllers. Arm motions are specified in joint space during walking, so no extra computation is needed for the arm joints.

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.

Sensing and State Estimation: Joint position is sensed by LVDTs on the actuators and velocity is the low-pass filtered finite difference of positions. Actuator force is measured using oil pressure in the piston chambers. Pelvis angular velocity and orientation are taken directly from an inertial measurement unit (IMU) mounted on the pelvis. Pelvis linear velocity and position are estimated using an Extended Kalman filter based on the IMU acceleration measurements and forward kinematics [13].

In addition to the pelvis estimator, we also implemented an extended Kalman filter that estimates the modeling error at the CoM level using linear inverted pendulum (LIPM) dynamics [14]. 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) [15], which is an offset to the current capture point. 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 (although earthquakes do happen at the DRC location in California), or external agents pushing on the robot. 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 (reset).

A derivation of the corrected capture point starts with LIPM dynamics augmented with a true center of mass offset and a true external force:

$$\ddot{c} = (c + c_{offset} + f_{ext} * z/mg - cop) * g/z = (c + \Delta - cop) * g/z \quad (1)$$

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 [15]:

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

For dynamic walking, the corrected capture point goes beyond the current single foot support polygon during swing phase and is captured by the touchdown foot. One way to predict a fall is to use the swing foot time to touchdown and predict if the current capture point is within the support polygon of possible or desired footholds. This approach is complicated because it depends on the controller. Our simpler solution is to use a heuristic that detects a fall only if the corrected capture point is outside of the current support polygon for a continuous period of time. The time threshold was set to 0.6 seconds after extensive testing. As soon as a fall is predicted, there are several things the humanoid can do, such as using angular momentum to maintain balance, or changing the current or future foot placements. We have implemented a simple step recovery controller that works in single support where the robot corrects the next foot placement and timing using a simple heuristic to avoid self collision. We would only step left if the robot is falling left, and similarly for right steps. We avoid crossing the legs in using foot placement to balance laterally.

Results of Safety Code: Robot caught on door frame: In the DRC rehearsal, the robot was caught on the door frame when sidestepping though (Figure 1). 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 2).

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 3), 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 3), 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.

Fig. 1. 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.

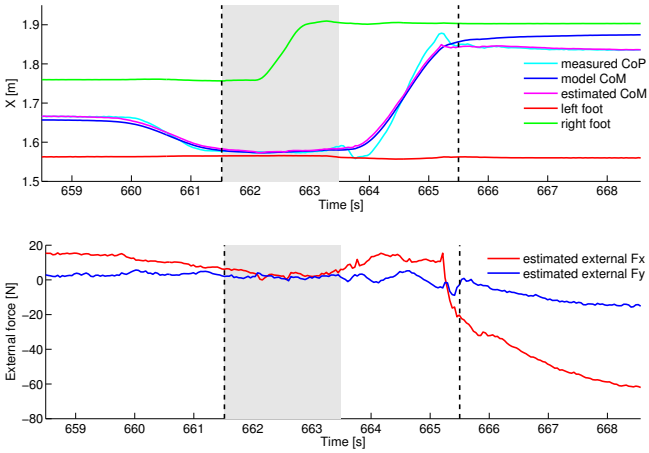

Fig. 2. Atlas was caught on the door frame when sidestepping through it during the DRC rehearsal. The walking controller delayed liftoff and remained in double support when the external force estimator detected a large change in the estimated external force in the robot's sideways direction (F_x, through 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.

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 3). The operator was then able to recover from an otherwise catastrophic scenario.

The time plot in Figure 3 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 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

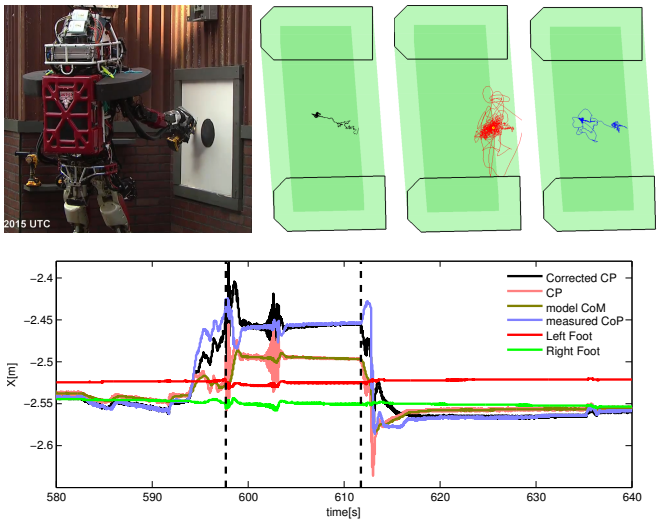

Fig. 3. Top Left: Robot posture after error detection. Top Right: Black trace: The corrected capture point (CCP) up to the error detection, Red trace: The CCP during the "frozen" period, and Blue trace: The CCP moves back to the center of the polygon of support during manual recovery. Bottom: Plots of 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.

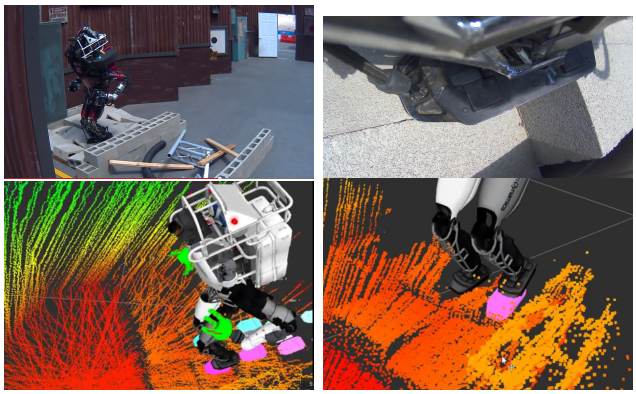

Fig. 4. Terrain Error: Top Left: The "frozen" robot, Top Right: A view of the left foot from the knee camera showing it straddling two cinder blocks. Bottom row: Mismatch between the plan and the actual step

that the robot is falling again, while the corrected capture point (CCP) moves towards the interior of the safe region.

Interface: 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.

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. 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 4). It was very impressive 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.

Perception: Page limits prevent us from fully describing our perception approaches and algorithms. We supported sensor fusion and object/environment mapping. LIDAR information was combined with stereo vision. The user could designate targets by using the mouse to designate parts of displayed images and point clouds, and our perception code supporting transforming this "scribble" into a useful object segmentation. Further details can be found in [16].

IV. Tasks: Strategies and Results

Driving: Figure 5 shows our robot driving the Polaris X900 vehicle in the DRC Finals. We won the "best driver" award in the DRC Trials [9]. 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 a 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 [17] 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 [18].

Egress: Our strategy for getting out of the car was to maximize contact with the car, as well as provide some mechanical aids (Figure 5). Maximizing contact led to using the car to stabilize the robot as much as possible. It was 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. This was an impressive but perhaps unnecessary demonstration of high quality robot control. MIT released a video of their robot standing on one foot while the vehicle is being shaken, which is a nice demonstration of high performance control but perhaps is overkill for this task [19]. Both IHMC and MIT added a handle to the car within easy reach of the robot, 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 our 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.

Fig. 5. Driving and egress (getting out of the car) tasks.

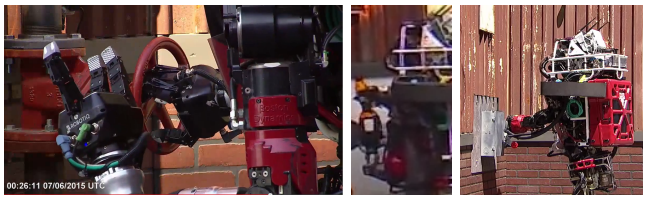

Fig. 6. Rotating the valve, reorienting the drill and regrasping it to turn it on, and having completed the "push the switch down"surprise task..

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 [20].

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 1). 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 [21].

Valve: The valve task involved approaching and turning a valve (Figure 6). 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.

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. This appeared to be an issue for other teams. MIT's drill bit came out of the wall during cutting on day 2 and the cut was not deep enough, causing the robot to fall when it tried to punch a hole in the wall [6].

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 with the new forearms 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. More details are available in [22]. 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.

Terrain and Stairs: We have already described our approach to controlling robot walking. It was crucial 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 7 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 [10]. More details on a paradigm shift are available from [23].

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 value.

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 response to different robot errors and external disturbances are often quite different and conflicting.

We could have done more to provide early fall predictions. In manipulation, we tracked the corrected capture point. In walking, the corrected capture point moves around quite a bit. Its deviation from its expected motion can be used to predict falling. 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.

Sensing is cheap, so let's use as much as possible. We strongly believe that humanoids should be outfitted with as much sensing as possible. The absence of horizontal force and yaw torque sensing in the Atlas feet limited our ability to avoid foot slip, reduce the risk of falling, and optimize gait using learning. We commissioned Optoforce to build true six axis foot force/torque sensors for the Atlas feet, but the sensors were not ready in time for the DRC. We will explore how much they improve performance in future work. Redundant sensor systems make calibration and detection of hardware failure much easier.

Super-human sensing is useful. Super-human sensing (whole body vision systems, for example) is a useful research area which could greatly improve humanoid robustness and performance. We used vision systems on the wrists to guide manipulation and on the knees to guide locomotion. Time prevented us from implementing planned vision systems located on the feet. We plan to build super-human feet with cameras viewing in all directions and IMUs (accelerometers and gyros). Our goal is to measure foot translational acceleration (accelerometers), linear and angular velocity (optical flow and gyros), and track foot translation, orientation, and relative positions (IMU orientation tracking and image matching). Longer term, we intend to build robust high resolution tactile sensing for the soles of the feet, as well as similar robust high resolution sensing skin. We intend to build many types of optical sensing into the robot's skin, to do obstacle and collision detection at a variety of distances and resolutions [24].

V. What Did We Learn From Our Own Work?

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 vary useful. The Atlas robot only allowed head nodding (rotation around a horizontal axis in the frontal plane (a pitch axis)).

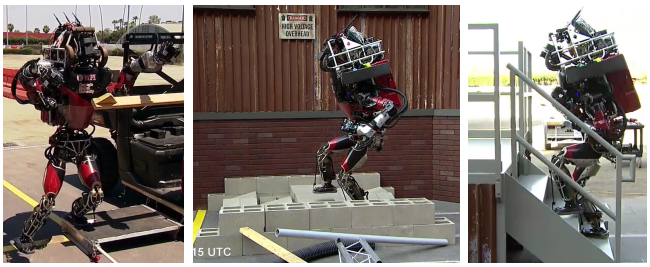

Fig. 7. Stepping off platform, Walking over terrain, Climbing stairs

Changing dimensionality is difficult for current control approaches. Our quadratic programming-based full-body control typically introduces large command discontinuities when the dimensionality of the problem changes due to contact changes, stick/slip change on any contact, singularities, and hitting or moving off of joint stops or limits. A typical step involves heel strike, foot flat contact, and toe push off. Walking on stairs may involve walking on the toes as well as on the sole of the foot. A large step down may involve jumping (no foot contacts), toe strike and foot flat, and the knee may go from straight to quite bent. Prioritized or constraint hierarchy approaches have similar problems with structural change of kinematics or dynamics.

Walk with your entire body. 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 [20], as was our ladder climbing in the DRC Trials [9].

Blend, don't switch. We developed an approach to structural change smoothing in our quadratic programming. In a previous implementation, the inverse dynamics QP changed dimension based on the number of contacts. Discrete changes can also happen due to constraint manifold changes such as during toe-off, when the foot frame CoP is constrained at the toe. Such changes will cause sudden jumps in outputs that can induce undesired oscillations on the physical robot. These jumps are caused by structural changes in the problem specification, and cannot be handled properly by just adding cost terms that penalize changes. Our solution is to maintain the same QP dimension and gradually blend the constraints over a short period of time. We always assume the largest number of contacts, but heavily regularize the magnitude of the contact force and relax the acceleration constraints for the fake contacts. Inequality constraint surfaces are changed smoothly when a change is required.

Design robots to survive failure and recover. 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. The Atlas robot was too top heavy and its arms

were too weak (similar to a Tyrannosaurus Rex) for it to reliably get up from a fall, especially in complex patterns of support and obstacles such as lying on the stairs, rough terrain, debris, or in the car or door frame. 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 [23]. 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 [23].

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.

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.

Dynamic approaches are often easier than static/quasi-static approaches. Although dynamic behavior is often more impressive to view, dynamic approaches are often easier than static/quasi-static approaches for the robot to actually do. However, perception, planning, and control must keep up with the higher movement velocities. It is easier to ride a bicycle at a higher velocity (within reason) than at a very slow velocity. In situations where footholds are plentiful (such as flat floors) and obstacles are sparse, dynamic locomotion approaches often provide more control authority and are more robust to modeling errors, due to the higher rate of foot placements (cadence). Static and quasi-static locomotion is limited by the ability to control center of pressure location under the stance foot in single support. Faster gaits don't need to carefully control center of pressure location under a foot, but simply place the foot appropriately. An example of this is that point feet (such as stilts) can often be used for dynamic gaits, but not for biped static gaits. We had a lot of trouble stepping up and down. In this case dynamic approaches can use momentum, and rely on the rear leg less, escaping kinematic constraints on the rear leg. Stepping down would involve falling to the next foothold instead of having control authority at all times. Jumping is an extreme case. We are currently investigating more dynamic approaches to the relevant DRC tasks.

Walking on real rubble involves making footsteps, not just finding footsteps. It turns out you have to plan more in terms of *making footholds* on terrains made up of aggregates (loose soil, gravel, pebble beaches or piles of rounded stones, sand, snow, stream beds, and realistic rubble) rather than *finding footholds.* You use your foot to compact a foothold so it remains solid when you put weight on it. In this case footstep planning needs to choose good places to make footholds, rather than good existing footholds.

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

Learning to plan better is hard. To be computationally feasible for real-time control, a cascade of smaller optimizations is usually favored, which for us and many others eventually boiled down to reasoning about long term goals using a simplified model and a one-step constrained convex optimization that uses the full kinematics and dynamics. Although effective, this approach suffers from the common problem that plagues hierarchical systems: how to generate a different high-level plan when something goes wrong at a lower level, which is not even modeled or represented in the high-level planner? We think the ideal solution is to include as complete a model as possible in the high-level planner and replan as often as possible, but this is unfortunately currently computationally impractical. A second alternative is to give limited foresight to the low-level controller and guide it towards the original high-level goal in that limited horizon. We have experimented with providing the approximated local value function computed by DDP to the inverse dynamics controller, but we have yet to observe a significant increase in either performance or stability. We think this is due to not previewing far enough into the future. For walking, we had the most issues with kinematic related constraints, such as rear ankle joint limits when taking a step down and knee singularity when it goes straight. Just adding inequality constraints on accelerations was not sufficient in most cases. In the end, we worked around these issues with special purpose heuristics that operate in joint space and incur increasingly higher costs as the system approaches these constraints. Although crude, it is a way of injecting some notion of the future into greedy local optimization. Some high-level changes are also necessary, such as limiting step length and allowing toe-off in various situations. A third direction is to insert a simpler trajectory optimizer that has a shorter horizon, but replans much more rapidly. We are currently exploring this idea by adding a receding horizon control component at the 1kHz time scale rather than at the 1Hz time scale, as is currently done.

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 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 [23].

VI. What Did We Learn At The DRC?

We are collecting data of what caused various events at the

DRC from all teams. We want solid data to test hypotheses about what led to success or problems in the DRC, including falls, failed attempts at tasks, long periods of robot inactivity, operator errors, etc. Many teams have already responded [4].

Did the time pressure or "speeding" lead to performance issues? We believe that no robot moved fast enough to have dynamic issues sufficient to cause it to lose control authority. However, it is clear from the data that trying to do the tasks quickly caused huge problems for the human operators. Of the six teams that have responded by the conference submission deadline, the top five each had major operator errors leading to a fall, a reset or large delay, or a task failure: (IHMC 2nd place, CHIMP 3, NimbRo 4, MIT 6, and WPI-CMU 7) [4]. After entering the contest believing that robots were the issue, we now believe it is the human operators that are the real issue. They are the source of the good performance of the DRC robots, but they are also the source of many of the failures as well. The biggest enemy of robot stability and performance in the DRC was operator errors [4].

The most cost effective research area to improve robot performance is Human-Robot Interaction (HRI). Developing ways to avoid and survive operator errors is crucial for real-world robotics. Human operators make mistakes under pressure, especially without extensive training and practice in realistic conditions. Interfaces need to help eliminate errors, reduce the effect of errors, and speed the recovery or implement "undo" operations when errors happen. Interfaces need to be idiot proof, require no typing and have no check boxes, minimize the information displayed, and the options the operator has to decide between. The best interface is perhaps a large **go** button and a much larger red **STOP** button.

Have we figured out how to eliminate programming errors? No. IHMC, MIT, and WPI-CMU all had bugs that caused falls or performance issues that were not detected in extensive testing in simulation and on the actual robots doing the DRC tasks in replica DRC test facilities [4]. IHMC in particular tried to have excellent software engineering practices, and still had an undetected bug that helped cause a fall on the stairs on day 1 of the DRC [7].

Are there other common issues? Many Atlas robots had faulty forearms that needed to be replaced, especially after a fall. Two handed drill task strategies turned out to be a big mistake. One failure mode was that Atlas forearm motors overheated and were automatically shut down. NimbRo also had motor overheating issues in their leg/wheel hybrid robot.

What did other top Atlas teams do about fall detection and prevention? MIT developed step recovery and bracing (using nearby surfaces for stabilization), but disabled them during car egress as its complexity and difficulty in doing good pose estimation relative to the car made finding reliable recovery strategies difficult [6]. We also disabled fall detection and recovery during egress for these reasons, and would have relied on manual intervention by an operator to recover from an egress failure. IHMC monitored the Capture Point [15] during manipulation tasks. The operator was alerted (by an audio beeping sound) if the Capture Point was off by about 2cm and the beeping increased in frequency as the error increased. If a threshold of 4cm was exceeded arm motions would be automatically stopped and the behavior queue reset. Both of these features prevented some falls when the robot was pushing too hard against a wall or fixed object. IHMC reports that this greatly reduced the operator load, as the operator did not have to worry as much about hitting things. An example is bumping a wall when picking up the drill. With these features, the operator does not need to preview motions and triple check for collisions. With an emergency abort, operators could be less cautions. In a situation where falls are not fatal (with the DRC reset or the ability to get back up) IHMC found that their robot walked faster and actually fell less often, because operator's attitudes changed [7].

Leg/wheel hybrids are very successful when flat floors and sparse light debris are present. CHIMP, KAIST, UNLV, RoboSimian, and NimbRo are examples of this. It is an open question what combinations of wheels, tracks, and limbs will be useful in the spectrum of rubble typically found in a collapsed building or an explosion or earthquake site.

Some leg/wheel hybrids and quadrupeds did not do any rough terrain. NimbRo, Aero, and RoboSimian did not do the rough terrain task or the stairs task at the DRC. It was possible to do the debris task instead of the terrain task and robots with wheels chose to do the debris task (including the Hubo teams). CHIMP also plowed through the debris, and did not do (or need to do) the rough terrain task. CHIMP did do the stairs. This pattern of not doing rough terrain in the form of tilted cinder blocks or stairs is surprising, in that tracked and quadruped designs were originally advocated as better ways to deal with rough terrain than bipedal designs. All of the biped robots that got that far chose the rough terrain task instead of the debris task. We found that shuffling through the debris was unreliable due to jamming, and picking up the debris was too slow.

Mechanical details of the robot could make a difference in the DRC tasks. In addition to wheels or tracks, mechanical details such as the ratio of the size of the feet to the center of mass height could make a difference in how hard the DRC was. A practical limit to the foot size was the size of a cinder block. No robot went that far in taking advantage of the details of the DRC course. We believe the Atlas robots had small feet relative to other bipeds, but we do not have foot measurements and center of mass heights for the other robots.

Wheeled vehicles need to balance and get un-stuck. To our surprise, many wheeled and tracked vehicles fell (CHIMP, NimbRo, Aero) or almost fell (KAIST). It was clear from the DRC that many of these robots were operating in the dynamic regime, and all needed to take dynamics into account to avoid roll-over, avoid getting stuck, or recover from being stuck.

Why didn't any robot use railings, walls, door frames, or obstacles for support and stabilization? Humans use stair railings, and brace themselves by reaching out to a nearby wall when walking over difficult rough terrain. Why didn't any DRC robots do this? Full body locomotion (handholds, bracing, leaning against a wall or obstacles) should be easier than our current high performance minimum contact locomotion approaches. Some robots used their arms to help during egress (KAIST, CHIMP, RoboSimian, WPI-CMU) [20]. We were one of the few teams to use arms in ladder climbing in the DRC Trials [9]. IHMC chose not to use the handrail on the stairs because during development the Atlas robots had very poor arm force control due to faulty wrist force/torque sensors, and the Atlas arms have a lot of static friction making implementing arm compliance difficult [7]. We had similar problems with our Atlas robot.

How to use foot force/torque sensors? Of the top three Atlas teams, our understanding is that IHMC and MIT both used the foot force/torque information just as a binary contact sensor [6], [7]. We used foot force/torque sensing in our state estimator and falling prediction, and controlled the robot using full state feedback and joint level force control, which includes joint (including ankle) torque feedback, but does not include direct contact force/torque feedback. Some ZMP-based robots use output feedback in the form of contact force/torque feedback to do foot force control. We need

to introduce output feedback and contact force/torque control to our QP and state-based full body control and perhaps to higher level control based on simple models.

Gentle touchdowns make walking easier or harder? Our footsteps seemed much softer with less of a shock wave travelling up the robot body than other Atlas teams. Does a firm footstep (a stomp) make sensing and control of locomotion easier or harder?

Is compliance useful? If so, how much, when, and where? We note that when Atlas robots fell, they often seemed very stiff with legs sticking up in the air and joints not moving during the fall (all of the videos of the other Atlas teams seem to have this behavior, similar to videos of the Honda Asimo falling). Perhaps very little compliance was actually being used in these so-called force-controlled robots.

Be ready for the worst case. In our Day 2 run we lost the DARPA provided communications between the operators and the robot in a full communication zone (standing before the stairs) for at least six minutes, which counted against our time. The lesson here is that for real robustness, we should have planned for conditions worse than expected. We could have easily programmed behaviors to be initiated autonomously if communications unexpectedly failed. Our understanding is that CHIMP lost communications during several tasks on day 2, including the driving task, when full communications were scheduled. CHIMP crashed the car, when it could have easily autonomously stopped, or autonomously driven the entire driving course.

There is something wrong with our field of humanoid robots. We have collectively produced many videos showing convincing humanoid robot performance. However, in a situation where the researchers did not control the test, most humanoid robots, even older and well-tested designs, performed poorly and often fell. It is clear that our field needs to put much more emphasis on building reliable systems, relative to simulations and theoretical results. We need to understand why our current hardware and software approaches are so unreliable.

VII. Conclusions

The "slow and steady" strategy did not win this race, but it came close. We implemented safety features in our controller to detect potential catastrophic failures, stop the current behavior, and allow remote intervention by a human supervisor. We implemented a human interface that allowed remote operators to usefully intervene and solve problems. We were the only team in the DRC Finals that tried all tasks, did not require physical human intervention, and did not fall in the two missions during the two days of the DRC Finals. We believe taking operator and robot error handling seriously is a key step towards reliable humanoid robot performance.

Acknowledgements

This material is based upon work supported in part by the DARPA Robotics Challenge program under DRC Contract No. HR0011-14-C-0011 and the US National Science Foundation (ECCS-0824077 and IIS-0964581).

References

[1] DARPA, "A celebration of risk (a.k.a., robots take a spill)," https://www.youtube.com/watch?v=7A_QPGcjrh0, 2015.

[2] IEEESpectrum, "A compilation of robots falling down at the DARPA Robotics Challenge," https://www.youtube.com/watch?v=g0TaYhjpOfo, 2015.

[3] DRC-Teams, "What happened at the DARPA Robotics Challenge?" www.cs.cmu.edu/~cga/drc/events, 2015.

[4] C. G. Atkeson, "Team WPI-CMU: Darpa Robotics Challenge," www.cs.cmu.edu/~cga/drc, 2015.

[5] WPI-CMU, "Team WPI-CMU DRC 2015," https://www.youtube.com/watch?v=BgfsxH3poCU, 2015.

[6] MIT, Personal communication, 2015.

[7] IHMC, Personal communication, 2015.

[8] Wikipedia, "Touchdown celebration," en.wikipedia.org/wiki/Touchdown_celebration, 2015.

[9] M. DeDonato, V. Dimitrov, R. Du, R. Giovacchini, K. Knoedler, X. Long, F. Polido, M. A. Gennert, T. Padir, S. Feng, H. Moriguchi, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Human-in-the-loop control of a humanoid robot for disaster response: A report from the DARPA Robotics Challenge Trials," *Journal of Field Robotics*, vol. 32, no. 2, pp. 275–292, 2015.

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

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

[12] W. Huang, J. Kim, and C. Atkeson, "Energy-based optimal step planning for humanoids," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, Karlsruhe, Germany, May 2013, pp. 3124–3129.

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

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

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

[16] B. P. W. Babu, X. Cui, and M. Gennert, "Robust sensor fusion and object feature extraction for humanoids," in *in preparation*, 2015.

[17] A. Geiger, "LIBVISO2: C++ Library for Visual Odometry 2," www.cvlibs.net/software/libviso/, 2012.

[18] K. Knoedler, VelinDimitrov, D. Conn, M. A. Gennert, and T. Padir, "Towards supervisory control of humanoid robots for driving vehicles during disaster response missions," in *IEEE International Conference on Technologies for Practical Robot Applications*, 2015.

[19] MIT, "Egress robustness tests," https://www.youtube.com/watch?v=F5CBRmDQXTk, 2015.

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

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

[22] B. P. W. Babu, R. Du, T. Padir, and M. Gennert, "Improving robustness in complex tasks for a supervisor operated humanoid," in *submitted to Humanoid Robots (Humanoids), 2015 15th IEEE-RAS International Conference on*, 2015.

[23] C. G. Atkeson, "Big Hero 6: Let's Build Baymax," build-baymax.org, 2015.

[24] V. Lumelsky, "Home page," http://directory.engr.wisc.edu/me/faculty/lumelsky_vladimir, 2015.