Full-body Motion Planning and Control for The Car Egress Task of the DARPA Robotics Challenge

Chenggang Liu[1], Christopher G. Atkeson[1], Siyuan Feng[1], and X Xinjilefu[1]

***Abstract*— We present a motion planning and control method for contact-rich full-body behaviors, particularly, the car egress task of the DARPA Robotics Challenge (DRC). We take advantage of human experience by manually specifying multiple task phases, each of which has a specific contact mode. Then, we optimize a sequence of static robot poses as well as contact locations that satisfy all constraints, such as geometric constraints, joint limit constraints, and equilibrium constraints. To accommodate model errors, uncertainties, and satisfy continuous contact constraints, we use online robot pose optimization to generate smooth trajectories based on sensor feedback and user inputs. We demonstrate with experiments that our motion planning method is capable of generating a rough plan for the challenging car egress task of the DRC. Combined with online robot pose optimization, the rough plan could be applied with excellent performance.**

I. INTRODUCTION

The DARPA Robotics Challenge (DRC) aimed to develop robot systems that are capable of assisting humans in responding to natural and man-made disasters. In the first task, the robot needs to drive a car, which is a Polaris Ranger X900, for a certain distance and then exit the car. The completion of this task demonstrates mounted mobility and the ability to operate vehicles designed for humans.

We are using the Boston Dynamics Atlas robot, as shown in Fig. 1. In this paper, we present our motion planning and control method for whole-body manipulation behaviors with many contacts, particularly, for the Atlas robot to get out of a vehicle.

(a) the Atlas robot (b) Polaris Ranger X900

Fig. 1: A photo of the Atlas robot and the vehicle, Polaris Ranger X900.

Although seemingly straightforward for humans, car egress turns out to be so difficult for the Atlas robot that no team succeeded in doing it or even tried it in the DRC Trials in December, 2013. Besides limited development time for each team, car egress on its own is difficult. First of all, humanoid robots are intrinsically floating base systems, which means there is no body part fixed to the ground of the vehicle. Therefore, the first consideration is to maintain balance. Second, the completion of full-body manipulation tasks relies heavily on the maintenance of contacts. For the car egress task, the free space is very limited, which makes motion planning challenging. Furthermore, the Atlas robot is not as flexible as humans which makes moving inside the car difficult. The Atlas robot also lacks observability, for example, it can't detect the location and magnitude of contact forces except for those on its feet and hands. In brief, just imagine how challenging it is for a fat person wearing a heavy backpack and a thick diving suit to exit a car.

The most popular motion planning technologies for humanoid robots are those based on probabilistic sampling due to their ability to plan efficiently in high-dimensional C-spaces [1], [2], [3], [4], [5]. However, pure rejection sampling doesn't work well when closed-chain and end-effector pose constraints exist because the volume of the feasible space is zero in the C-Space and other constraints, such as equilibrium and collision-avoidance, further reduce its volume. To address the problem of sampling on constraint manifolds, approaches using projection operations have been developed [6], [7], [8].

Another widely used technique to generate full-body motions for humanoid robots is based on inverse kinematics and dynamics. A hierarchical framework designed for humanoid robots is presented in [12], [13]. It handles constraints and multiple objectives by solving a hierarchical QP problem. However, it is hard to determine the priorities among several objectives. Moreover, it may suffer from singularities because the QP problem linearizes the system about the model operating point and doesn't take long term cost into account. Some approaches first optimize task-space trajectories, such as trajectories for the center of mass and a swing foot, using a simplified model, such as the linear inverted pendulum model. Then they solve inverse kinematics to generate full body motions [9], [14]. However, in a very constrained environment, it is hard to find a feasible full-body trajectory that is collision-free and satisfies equilibrium constraints and task-space trajectories.

During motion planning, we take advantage of human experience by manually defining multiple task phases, each of which has a specific contact mode. We then specify shape objectives, such as the upper body and foot orientation,

[1]Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsbugh, USA `cgliu2008@gmail.com, cga@cmu.edu, sfeng@cs.cmu.edu, xxinjile@cs.cmu.edu`

geometry constraint cost and so on. After that, we optimize a sequence of robot poses/contact locations that can maintain balance. To satisfy constraints continuously and accommodate approximation and other errors, we use the offline generated robot poses as desired poses for each task phase and solve an inverse kinematics problem on the fly based on sensor feedback and user inputs.

The organization of this paper is as follows: Section II presents the motion planning method and the full-body control method. Experimental results are shown in Section III, followed by discussions and conclusions in Section IV.

II. Approach

A. Motion Planning

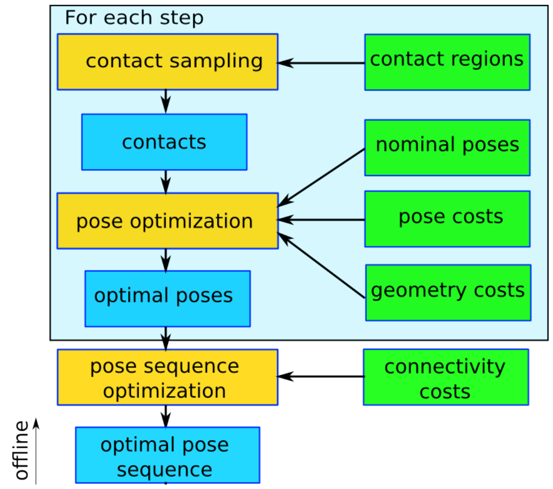


Fig. 2: Overview of the offline part of our motion planning approach. Green boxes are user input data. Yellow boxes are procedures. Blue boxes are intermediate data.

For the sake of clarity, similar to [10], we denote a contact as an association between a point on the robot and a point in the environment. A set of contacts forms a candidate 'stance', σ. Because the Atlas robot has a similar body configuration as humans, for a specific full-body manipulation task, we take advantage of human experience by manually specifying multiple task phases, each of which has predetermined body contacts. We then optimize contact locations in the environment and robot poses for each task phase. For example, we specify the following task phases for car egress: Phase 0) *sit*, Phase 1) *right foot out*, Phase 2) *stand up inside*, Phase 3) *move right inside*, Phase 4) *left foot out*, and Phase 5) *stand up*. For each phase, we specify desired contact locations or desired contact regions on the car. If a contact region is specified, contact locations are sampled in order to find the best contact locations. We use 2D Halton sampling to allow us to sample more contacts with low discrepancy as needed [5]. We also specify nominal robot poses for the optimizer to start with and shape/geometry costs in order to generate natural poses without unexpected collisions. The offline part of our motion planning approach is shown in Fig. 2. The reason for us to use contact sampling is to reduce the computational cost

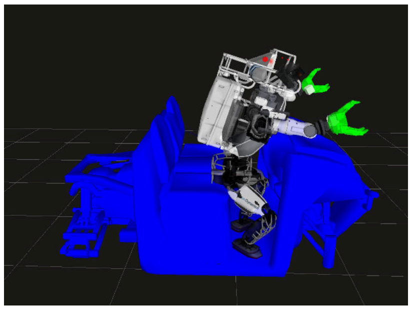


Fig. 3: User Interface for nominal pose selection. The position and orientation of the hands, the feet, the pelvis, and the torso can be controlled interactively through this UI.

from a continuous contact location optimization to a discrete contact location optimization. Moreover, contact sampling allows us to easily take advantage of parallel computing.

After creating multiple optimal robot poses for each phase, we search for the optimal sequence of robot poses as well as the optimal contact locations. We consider a 'weak' connection between two stance manifolds using a connectivity cost rather than require that there exists a nonempty common set of two stance manifolds for them to be connected as in [10]. In our weighted graph, nodes are feasible configurations associated with a pose cost and edges are connections between two nodes associated with a connectivity cost. We find the optimal sequence of robot poses by finding the shorted path in the weighted graph.

1) Pose Optimization: Starting from a nominal pose, we use online optimization to enforce the contact constraints and find an optimal robot pose that is statically stable and without unexpected collisions.

Assuming there are n contact points for the current phase, to find a feasible robot pose, we solve a Nonlinear Programming Problem (NLP):

$$\begin{aligned} \underset{q,\tau,f_1,\dots,f_n}{\text{minimize}} \quad & C_p(q,\tau,f_1,\dots,f_n) \\ \text{subject to} \quad & C_\sigma(q) = 0 \\ & h(q) = B\tau + \sum_{i=1}^{n} J_i^\mathsf{T} f_i \\ & f_{i_{min}} \le f_i \le f_{i_{max}} \\ & q_{min} \le q \le q_{max} \\ & \tau_{min} \le \tau \le \tau_{max}, \end{aligned} \tag{1}$$

where τ is a vector of joint actuation torques, f_i is a vector of contact forces on the i^{th} contact point, $C_p(\cdot)$ is the pose cost function, $C_\sigma(\cdot)$ is a vector of distances between the points on the robot and their desired contact locations in the environment, $h(q)$ is the gravitational forces, B is a selection matrix, which is an identity matrix except for the top six rows that corresponding to the six DoFs of the floating base which are zero. J_i^T is the Jacobian matrix for the i^{th} contact point. We simplify the friction cone constraints with box constraints. We call the first constraint equation “contact constraints” and the second “equilibrium

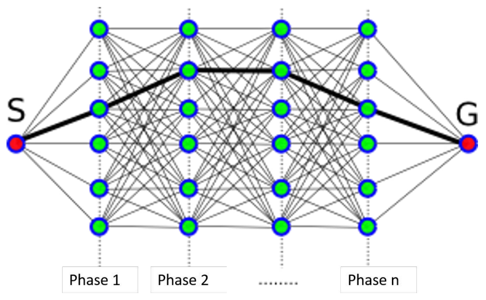


Fig. 4: An illustrative weighted graph. The green bubbles are optimal robot poses for each phase. The optimal sequence of robot poses is on the shortest path of the weighted graph, for example, the bold solid line.

constraints", hereafter. Because this nonlinear optimization problem has sparse gradients, we solved it using SNOPT [15].

However, without a 'good' nominal pose, we find this NLP is still hard to solve using SNOPT. Instead of taking joint torques as optimization variables, we remove them from this NLP and optimize q and f_i first, namely, only use the top six equations corresponding to the floating base in the equilibrium constraints. After optimization, we calculate τ according to the rest of the equations in the equilibrium constraints to see whether they are within torque limits. Most of the time, the resultant τ are within their limits. If not, we modify the pose cost function and re-optimize. For example, we penalize the pitch angle of the torso in order to reduce the actuation torque of the torso back joint.

For collision avoidance, we add geometry costs to the pose cost with special attention to the knees, the elbows and the chest because these are the parts most likely to have collision with the car. We use a smooth hinge-like function to steer these body parts away from collisions. The hinge-like function is zero for all positive distances and greater than zero for all negative distances and grows quadratically:

$$r(x) = \begin{cases} 0 & x \geq 0 \\ x^2 & x < 0 \end{cases} \tag{2}$$

We also add a shape cost to the pose cost, such as a cost on orientations of the upper body, the pelvis, the feet and the deviation from the nominal pose. The use of nominal poses and shape costs bias Pose Optimization towards natural poses.

There are several ways to get nominal poses. For example, it can be done by measuring the joint angles of a human demonstrator or placing the Atlas robot in the car. One efficient way that we used is to select a reasonable pose through an interactive User Interface we have developed as shown in Fig. 3. For a better result, multiple nominal poses can be used, which in general results in different poses after Pose Optimization.

2) Pose Sequence Optimization: To enforce the necessary condition, we define a connectivity cost between two successive robot poses, q and q', as the sum of 1) the penalty on the displacement of specified contact points on the body and 2) the penalty on the displacement of all joint angles:

$$C_c(q,q') = w_1\big(\sum_i ||p_i(q) - p_i(q')||^2\big) + w_2||B(q-q')||^2 \tag{3}$$

where p_i is the i^{th} specified contact point on the body and B is the same selection matrix as in Eq. (1).

We construct a weighted graph as shown in Fig 4 where the nodes are optimal robot poses associated with a pose cost C_p and edges are associated with a connectivity cost C_c. Dynamic Programming provides an efficient way to find the 'shortest' path (a path with the minimal total cost) among these nodes [16]. To take advantage of parallel computing, we use a value iteration method and the value update rule is given by:

$$V(q) = C_p(q) + \min_{q'} \{C_c(q,q') + V(q')\}, \tag{4}$$

where $V(q)$ is the value associate to robot pose q. Since all nodes have the same update rule, we update all nodes in parallel until their values converge.

Given pose q, the next adjacent pose q' in the optimal pose sequence is given by

$$q' = \arg\min_{q'} C_c(q,q') + V(q'). \tag{5}$$

B. *Full-body Control*

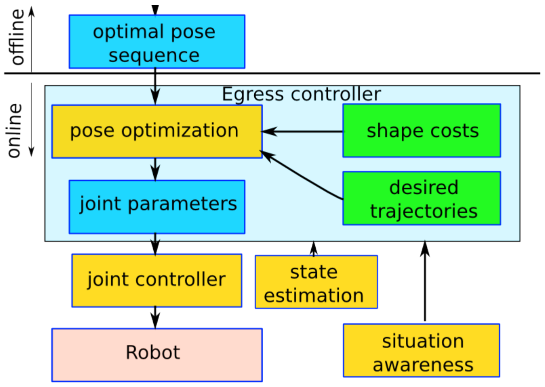


Fig. 5: Overview of the online car egress controller. Green boxes are user input data. Yellow boxes are procedures. Blue boxes are intermediate data.

The structure of the online full body controller is shown in Fig. 5. The kernal is an on-line pose optimization solver. It takes the joint position limits as inequality constraints and calculates robot pose change rate $\dot{q}$ by solving a Quadratic Programming problem:

$$\begin{aligned} \underset{\dot{q}}{\text{minimize}} \quad & \sum_i w_i||J_i(q)\dot{q} - k_p^c(x_{d_i} - x_i) - k_d^c(\dot{x}_{d_i} - \dot{x}_i)||^2 \\ & + w_d||B\dot{q}||^2 \\ \text{subject to} \quad & q_{min} \leq \dot{q}\Delta T + q \leq q_{max} \end{aligned} \tag{6}$$

where $J_i(q)$ is the Jacobian matrix, x_{d_i} and x_i is the desired and the calculated vector of body point position or joint angles, and B is the selection matrix. The desired joint angles are derived from the interpolation of the offline generated optimal pose sequence given by II-A.2. The output robot pose is given by the integration of the calculated robot pose change rate:

$$q = q + \dot{q}\Delta T, \tag{7}$$

where ΔT is the control cycle period. The joint parameters are given by $\theta_d = Bq$ and $\dot{\theta}_d = B\dot{q}$, where B is the selection matrix.

The hydraulic servo valves are controlled by on-board servo valve controllers that read sensor data and update the valve commands at 1kHz. The servo valve controllers compute valve commands, cmd, according to the following equation:

$$cmd = K_p(\theta_d - \theta) + K_d(\dot{\theta}_d - \dot{\theta}) + K_f(\tau_d - \tau), \tag{8}$$

where K_p is the gain matrix for joint angles, K_d is the gain matrix for joint angular velocities, and K_f is the gain matrix for joint actuation torques. For most of the time, we use PD control for all joints, namely, K_f is zero. In order to get soft touchdown of a foot, we first tilt the foot up, and then set the desired ankle torques to be zero and the corresponding gains in K_f to be non zero before touchdown. This results in soft touchdown and assures the foot is compliant with the ground surface.

The state estimator for the pelvis position and velocity is based on [17]. For the Atlas robot, there is a six-axis inertial measurement unit (IMU) attached to the pelvis link. The IMU measures angular velocity and linear acceleration, and it also provides an estimate of the pelvis orientation in the world frame. We use the orientation estimate from the IMU without modification. The pelvis position and velocity estimator is a multiple model Kalman filter based on reference points, namely, points that are considered to be stationary in the world frame. We design a steady-state Kalman filter for three candidate reference points, which are the left toe, the right toe and the pelvis, for the car egress task. Our heuristic to choose which contact point is stationary is as follows. When the total upright contact forces on both feet is less than half the total weight of the robot, we use the pelvis as the reference point. Otherwise, we use the toe with a larger upright contact force as the reference point.

The overall egress controller is a linear state machine and progresses state to state except for abnormal situations. So far, situation awareness is done by the operator. We installed a webcam on each knee, through which the operator is able to know the relative position of each foot w.r.t. the car and can change the lifting foot's trajectory on the fly. The operator can also look through the front camera to know about the relative position of the chest w.r.t. the dashboard and a horizontal bar we installed. The camera is automatically selected based on the state. The UI also shows the robot pose tracking which allows the user to be aware of abnormal situations and determine whether to intervene by manually controlling the position/orientation of a specific body part, such as the hip.

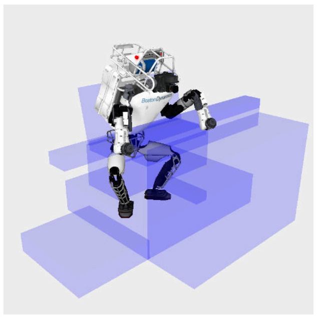
III. Results

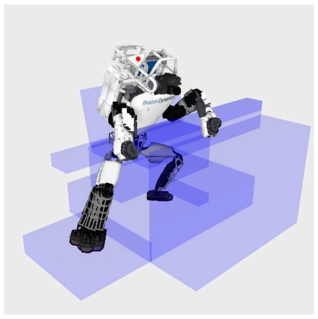
We provided several mechanical aids for egress in the DRC. Because Atlas was too big to sit behind the steering wheel, it had to exit the car from the passenger side, so we extended the accelerator pedal, which then became another obstacle the robot couldn't see during egress. Because of the kinematic limitations the Atlas robot has, such as the leg lengths, teams using the Atlas for egress installed an egress platform, such as Team MIT, Team IHMC, Team TRAC LABS, and Team WPI-CMU. One exception is that Team TROOPER had their robot slide out of the vehicle on pivoting metal rails. We installed a jig in the place of a seat cushion to fix the hip in order to increase repeatability and reliability, as the bottom of the pelvis was rigid, small, and had little friction with our hard seat, so the robot could easily slip and rotate. We also installed a horizontal wooden bar on top of the dashboard to replace the support a car door (which was missing) would have provided and provide additional contact opportunities during the entire behavior. We derive the dynamics/kinematics parameters of the Atlas robot from its Gazebo model [18] and parameters of the car based on its modification. We then use SDFAST [19] to generate the dynamics model of the robot for offline and online pose optimization.

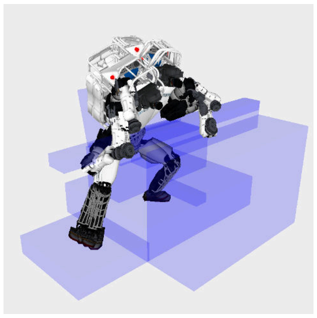
After the driving task, the robot needs to move from the final pose for driving to the initial pose for car egress. This is done by tracking robot joint trajectories. Because the robot always starts from the same location inside the car because of the seat jig, this open-loop control is very reliable.

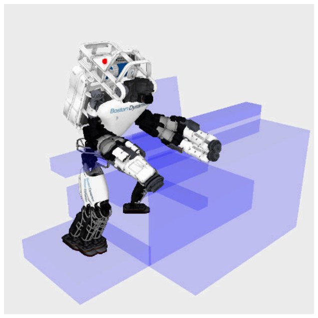
Starting from the initial pose of Phase 0, we generate poses for Phase 1) *right foot out* with contacts on the hip, both feet and both wrists; Phase 2) *stand up inside* with contacts on the chest, both feet and both wrists; Phase 3) *move right inside* with contacts on the chest, both feet and both arms; Phase 4) *left foot out* with contacts on the chest, both feet, and both arms; and Phase 5) *stand up* with contacts on both feet. We optimize the placement of the right toe by sampling contact locations in a 0.2×0.2 square meters' region on the egress platform for Phase 1-4. The pose costs, C_p, for Phase 1-4 are the same. It penalizes the displacement of the upper torso orientation, the pelvis orientation, feet orientation, and joint angles from the nominal pose. It also penalizes the external forces on the hands. The geometry costs for Phases 1-4 limit the left knee's position in the forward direction and the elbow's position in downwards direction. The result of Pose Optimization for each phase is shown in Fig. 6. For pose sequence optimization, w_1 is 10^4 and w_2 is 10^1. After value iterations of Dynamic Programming, the optimal pose sequence is shown in Fig. 7.

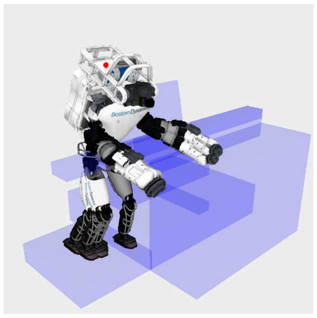
For egress control, w_d is 0.005 and others weights and gains used by online pose optimization are listed in Table I. We designed a swing trajectory and soft touchdown control for the right foot for the transition from Phase 0 to Phase 1; right foot rotation control (toe down, rotate, toe up) at the end of Phase 2; a swing trajectory and soft touchdown control for the left foot for the transition from Phase 3 to Phase


(a) Phase 0


(b) Phase 1


(c) Phase 2


(d) Phase 3


(e) Phase 4

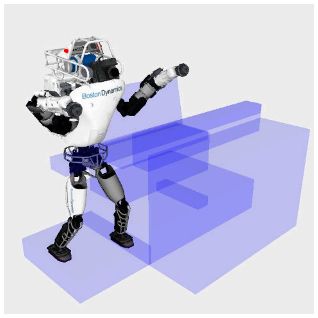

(f) Phase 5

Fig. 6: Optimal poses of each phase. Figs. 6a and 6f show the initial and the final robot pose. For Phases6b-6e, 50 candidate contact locations for the right toe are sampled and 50 robot poses are generated (10 of which are shown).

TABLE I: Weights and gains used by online pose optimization

	w	k_p^c	k_d^c
joint angle	0.05	2.0	0.9
pelvis position	0.05	4.0	0.9
foot position	500.0	4.0	0.9
hand position	50.0	4.0	0.9
com position	5.0	2.0	0.9
torso orientation	500.0	2.0	0.0

4; and a swing trajectory for the right foot and a transition motion to remove the chest contact at the end of Phase 4. We use a 3-axis force/torque sensor on each foot to measure upward ground reaction forces. The online optimization runs at 1000 Hz for real-time control and the sensors are sampled at the same frequency. To accommodate uncertainties during the competition, the controller allows the operator to adjust the right foot position before it gets out of the car and the left foot position before it lands on the egress platform based on video feedback from the knee cameras.

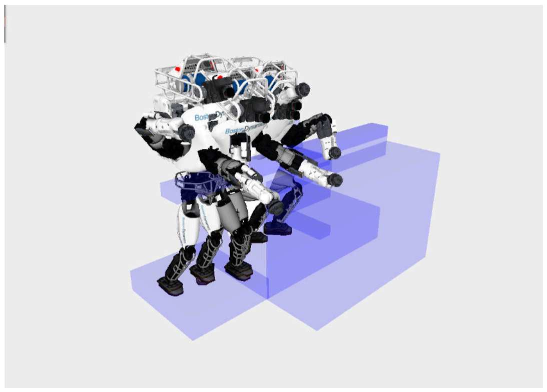


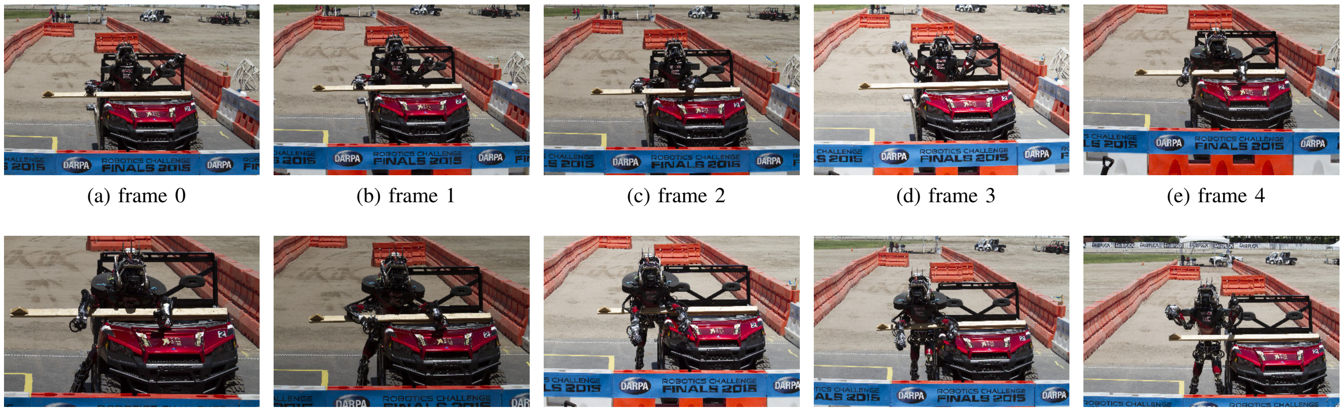
Fig. 7: The optimal pose sequence after value iterations of Dynamic Programming.

The robot pose tracking result from the DRC Finals on day 1 is shown in Figs. 8 and 9. The experiment video can be found at [20].

IV. Discussions and Conclusions

In this paper, we combine an offline robot pose sequence/contact location optimization for motion planning and an online robot pose optimization for full-body control. The former takes the task-level goal into account and finds an optimal sequence of static robot poses as well as contact locations. The latter takes the optimal sequence as input and adapts the plan according to sensor feedback and user inputs. The proposed approach can be applied to contact-rich full-body behaviors. Particularly, we presented car egress task results from the DRC Finals. The offline generated robot poses steer the online pose optimization away from singularities and simplify its cost functions. The online robot pose optimization enables us to accommodate model errors and uncertainties. As a result, we can achieve reliable performance. This approach worked four out of four times during the battery test, DRC Finals rehearsal, day 1, and day 2. It also worked 8 out of 10 times in the last 10 tests we did right before the DRC Finals (one failed because of deformation of the chest plate and the other failed because of an incorrect installation of the horizontal bar).

For contact-rich full-body behaviors, most of the time it is hard to avoid unexpected collisions. In our approach, we intentionally add more contacts, such as contacts on the chest and the arms, to enlarge the stable margin, which enables us to achieve statically stable equilibrium all the time. 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. During our car egress task, unexpected collisions occurred between the left knee, the left foot and the car, however, our controller accommodated these unexpected collisions well. For us, more contacts provided more stability.



(a) frame 0 (b) frame 1 (c) frame 2 (d) frame 3 (e) frame 4

(f) frame 5 (g) frame 6 (h) frame 7 (i) frame 8 (j) frame 9

Fig. 8: Robot pose tracking result from the DRC Finals on day 1. Frames 0-4 show the Atlas robot moved from the final pose of the driving task to the initial pose of the car egress task. Frames 4-9 show the robot pose tracking result of Phases 0-5 in Fig. 7.

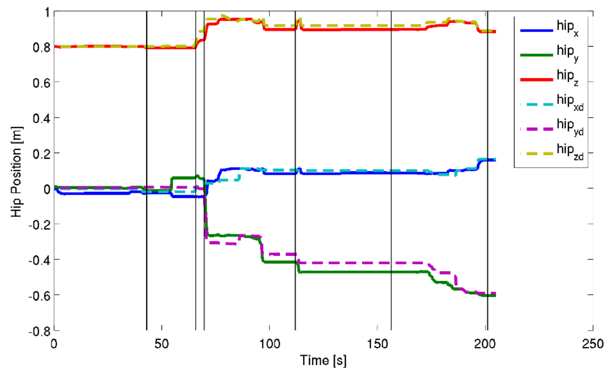


Fig. 9: Robot pose tracking result in term of the hip position from the DRC Finals on day 1. The six vertical lines from the left to the right correspond to the robot poses of Phase 0-5, respectively. The X axis is the forward direction, Y points to the robot's left, and Z points upward. The solid lines are from state estimation and the dashed lines are desired values from the controller.

ACKNOWLEDGMENT

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] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[2] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2, 2000, pp. 995–1001.

[3] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Motion planning for humanoid robots," in *Robotics Research. The Eleventh International Symposium*. Springer, 2005, pp. 365–374.

[4] J. Kuffner, JamesJ., S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-stable motion planning for humanoid robots," *Autonomous Robots*, vol. 12, no. 1, pp. 105–118, 2002.

[5] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[6] M. Stilman, "Task constrained motion planning in robot joint space," in *Intelligent Robots and Systems, IROS. IEEE/RSJ International Conference on*, 2007, pp. 3074–3081.

[7] D. Berenson, S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning on constraint manifolds," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, 2009, pp. 625–632.

[8] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[9] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based full body control for the DARPA Robotics Challenge," *Journal of Field Robotics*, vol. 32, pp. 293–312, 2015.

[10] K. Hauser, T. Bretl, and J.-C. Latombe, "Non-gaited humanoid locomotion planning," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, 2005, pp. 7–12.

[11] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots," in *Algorithmic Foundation of Robotics VII*, ser. Springer Tracts in Advanced Robotics, S. Akella, N. Amato, W. Huang, and B. Mishra, Eds. Springer Berlin Heidelberg, 2008, vol. 47, pp. 507–522.

[12] L. Sentis, "Compliant control of whole-body multi-contact behaviors in humanoid robots," in *Motion Planning for Humanoid Robots*. Springer, 2010, pp. 29–66.

[13] L. Sentis and O. Khatib, "A whole-body control framework for humanoids operating in human environments," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 2641–2648.

[14] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with simple dynamics and full kinematics," in *IEEE-RAS International Conference on Humanoid Robots*, 2014.

[15] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Journal on Optimization*, vol. 12, pp. 979–1006, 1997.

[16] R. Bellman, *Dynamic Programming*. Dover Publications, 2003.

[17] 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*, 2014, pp. 195–201.

[18] [Online] http://gazebosim.org/

[19] [Online] http://support.ptc.com/support/sdfast/index.html

[20] [Online] https://youtu.be/I8ONUwkOlMQ