Robust Ladder-Climbing with a Humanoid Robot with Application to the DARPA Robotics Challenge

Jingru Luo[1] Yajia Zhang[1] Kris Hauser[1] H. Andy Park[2] Manas Paldhe[2] C. S. George Lee[2]
Michael Grey[3] Mike Stilman[3] Jun Ho Oh[4] Jungho Lee[5] Inhyeok Kim[5] Paul Oh[6]

***Abstract*— This paper presents an autonomous planning and control framework for humanoid robots to climb general ladder- and stair-like structures. The approach consists of two major components: 1) a multi-limbed locomotion planner that takes as input a ladder model and automatically generates a whole-body climbing trajectory that satisfies contact, collision, and torque limit constraints; 2) a compliance controller which allows the robot to tolerate errors from sensing, calibration, and execution. Simulations demonstrate that the robot is capable of climbing a wide range of ladders and tolerating disturbances and errors. Physical experiments demonstrate the DRC-Hubo humanoid robot successfully mounting, climbing, and dismounting an industrial ladder similar to the one intended to be used in the DARPA Robotics Challenge Trials.**

I. Introduction

Robots will need robust ladder- and stair- climbing skills to navigate and perform maintenance tasks in man-made structures like homes, offices, and industrial environments. These behaviors are challenging to be implemented on humanoid robots because they require full-body coordination, fine motor skills for docking hands and feet, and substantial upper body strength. Other climbing robots have used specialized equipment or custom-designed environments (e.g., [1]–[4]). Motivated by the DARPA Robotics Challenge (DRC), we address multi-purpose humanoid robots that can also perform tasks that might be expected of a human aid worker in disaster scenarios, e.g., rough-terrain locomotion, manipulation, and driving. We apply this work to the DRC-Hubo humanoid (see Fig. 1) but our approach is adaptable to many humanoid robots and climbable structures.

Ladder-climbing is usually straightforward for humans, and can be broken into a sequence of sub-tasks, e.g., move the foot to a higher rung, transfer the center of mass to the supporting foot, raise the hand to a higher rung, etc. While the choice of contacts and motions in each sub-task may be intuitive for a human, a robot often has to generate its motion *de novo* due to subtle differences in kinematics and dynamic capabilities. As a result, designing effective robot climbing strategies is surprisingly challenging. For example, although humans usually face forward while climbing, our design experiments determined that backward climbing gives larger clearance between DRC-Hubo's legs and the ladder rungs, because its ankles are rather large and it cannot bend forward at the waist (see Fig. 2). This does not pose a sensing problem for DRC-Hubo because it can swivel its neck 180°. Another example is the use of a toe joint: human climbers dynamically toe-off when lifting a foot onto a higher rung, which reduces stress on the quadriceps which would otherwise need to lift the entire body weight, essentially performing a difficult one-legged squat. DRC-Hubo, on the other hand, has no toe joint, but has sufficient torque to lift itself easily on one leg.

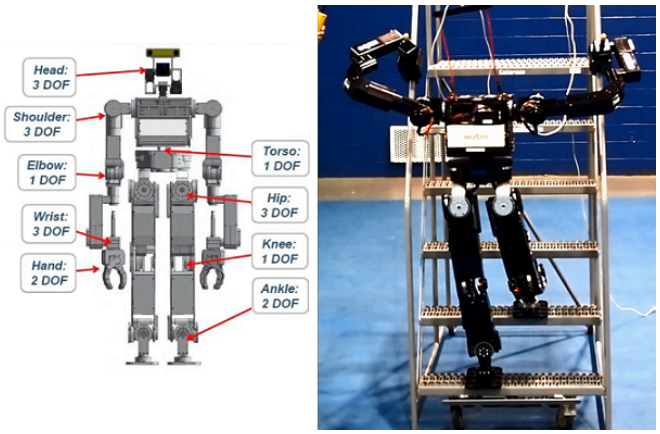

Fig. 1. Left: DRC-Hubo robot and its degrees of freedom. Right: DRC-Hubo climbing an industrial ship ladder using our approach.

This paper presents a planning and control framework for autonomous ladder climbing (Fig. 3) once the robot has acquired a model of its local environment. It has three important characteristics: 1) the use of motion primitive-based motion planning to give "hints" to the robot while remaining adaptable to different ladders [5], 2) generation of strictly feasible motions that obey physical constraints and robot performance limitations [6], [7], and 3) selective compliance to let the robot act stiffly in task-relevant dimensions and softly in the most uncertain dimensions [3].

The method plans climbing motions in seconds, and simulation experiments were employed to verify that the

*This work was supported by the Defense Advanced Research Projects Agency (DARPA) award # N65236-12-1-1005 for the DARPA Robotics Challenge and NSF grant IIS #1218534. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

[1]School of Informatics and Computing, Indiana University Bloomington, USA {luojing,zhangyaj,hauserk}@indiana.edu
[2]School of Electrical and Computer Engineering, Purdue University, USA {andypark,mpaldhe,csglee}@purdue.edu
[3]School of Interactive Computing, Georgia Institute of Technology, USA {mxgrey,mstilman}@cc.gatech.edu
[4]Korea Advanced Institute of Science and Technology, South Korea jhoh@kaist.ac.kr
[5]Rainbow Co., South Korea jungho77@rainbow.re.kr, inhyeok@kaist.ac.kr
[6]Mechanical Engineering and Mechanics, Drexel University, USA paul@coe.drexel.edu

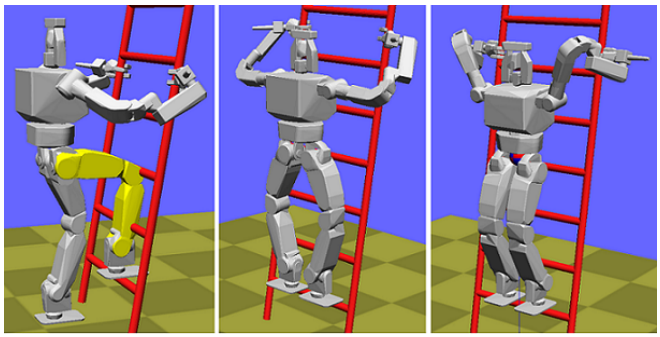

Fig. 2. Comparison of different climbing strategies. Left: in forward climbing, the leg joints can easily collide with the rungs (the colliding joints is drawn in yellow); Middle: in forward splayed-feet climbing, the leg clearance is improved but this strategy is not able to climb ladders near vertical; Right: backward climbing provides the best leg joints clearance.

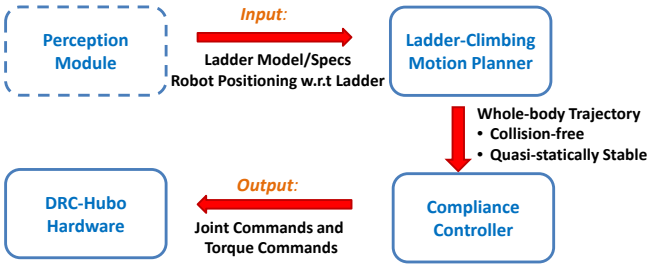

Fig. 3. Ladder-climbing framework. We assume a perception module (which is underdevelopment and not covered in this paper) provides the ladder environment as input to the ladder-climbing motion planner. The planner generates a feasible trajectory for a compliant controller which enables compliance on arm joints for error compensation.

robot can execute them successfully on a wide variety of ladders, including A-frame ladders, vertical ladders, and ship ladders. The system is also relatively robust. Experiments on the physical robot suggest that errors in rail position be tolerated as long as the rail is within the open fingers' range, approximately ± 5 cm left and right. Simulation experiments also suggest that errors of ± 2 cm in rung spacing may be tolerated in successful. Finally, we employed the system to enable the physical DRC-Hubo to climb up stairs while holding handrails, and to mount, climb, dismount a ship ladder similar to the ladder intended to be used in the DARPA Robotics Challenge Trials in December 2013 [8].

II. Integrated Hardware and Software Framework

This section summarizes our integrated planning, control, and execution framework. The major components will be described in more detail in later sections.

A. DRC-Hubo Robot

The DRC-Hubo humanoid is an upgraded version of Hubo-II+ built by Rainbow Co. for Team Hubo participating in the DARPA Robotics Challenge. This robot is about 140cm height and 50kg weight and has 33 degrees of freedom (DOFs): 7 in each arm, 6 in each leg, 1 in the waist, 3 in the neck, 1 in each hand for controlling the opening and closing and one extra for the trigger finger on the right hand(see Fig. 1).

The sensor head installed on DRC-Hubo contains a lidar and stereo cameras for perceiving the environment and providing visual feedback. The robot is also equipped with 3-axis F/T sensors on each wrist and ankle, accelerometers and gyros on the ankles and an inertia measurement unit for sensor feedback in motion execution. Two computer modules are located in the chest; one for real-time whole-body control through Controller Area Network (CAN), and the other for vision processing. In the current work, we do not use vision except to help a human operator pre-survey the ladder model. Work to integrate perception with planning is ongoing.

B. Ladder and Environment Modeling

The input to our system is a ladder model and relative pose of the ladder w.r.t. the robot. The output is raw motor commands for the DRC-Hubo to execute. It supports a wide variety of ladder models, including A-frame ladders, regular ladders, and ship ladders. The ladder specification contains an arbitrary number of stringers, rungs, and hand rails. Parameters describe the ladder inclination, number of rungs, rung spacing, rung width, rail height and cross-sectional geometries which can be circular or rectangular (Fig. 4). These models can be estimated in a number of ways, such as on-board perception, chosen by a human operator, or pre-surveyed from architectural blueprints. Obstacles like support rails or safety cages may also be incorporated into the environment, and the planner will avoid collision with these obstacles.

The framework also supports partially-known environments. When the ladder is only partially in-view, the robot will climb a user-specified number of steps, and replanning will be invoked once it gets a better view. This capability is particularly useful when the top of the ladder and/or dismount area is occluded from the robot's initial pose.

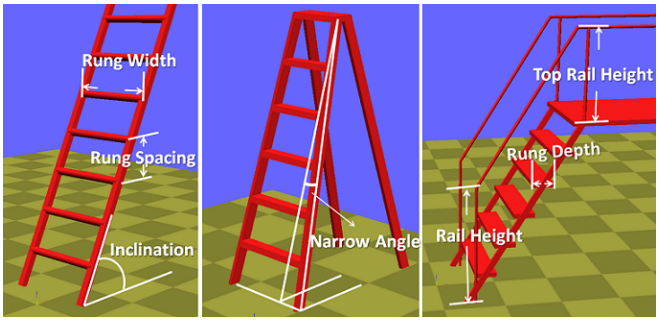

Fig. 4. Ladder models that can be addressed by our system. Left: a regular ladder. Middle: an A-Frame ladder. Right: a ship ladder. Basic parameters shared by all ladders are marked on the regular ladder and special parameters for A-Frame ladders and ship ladders are marked separately.

C. Motion Primitive-Based Planner

The planner takes the robot's posture and environment knowledge as input, and outputs a collision-free and quasi-statically stable joint-space trajectory. *Motion primitives*,

trajectories designed by human experts, provide the robot with prior knowledge about how to solve the ladder climbing problems – where the limbs are placed, how to keep balance, how to avoid collision with rungs, etc. Our planner adapts primitives to new ladders using a combination of optimization and sampling-based planning techniques [5]. To ensure reliable planning, it is important that the planner can backtrack rather than commit to all of its choices, because an early bad choice of contact can lead to stalled progress later on. Surprisingly, the planner can solve a wide variety of ladders quickly with only *one* primitive sequence.

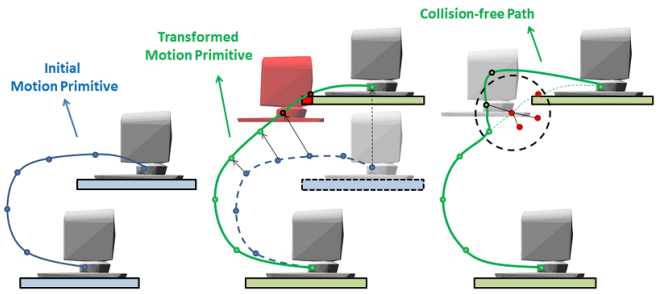

Fig. 5. Illustration of motion planning process. Left: A motion primitive includes limb placement and waypoints information designed for certain ladder. Middle: The motion planner first determines the limb placement in the new ladder, then performs a linear transformation that transforms the existing motion primitive in configuration space to fit the new environment. Right: A sampling-base process is used to retract the collision section (drawn in red) to obtain a collision-free path.

D. Compliance Controller

The controller interpolates the planned trajectory and sends commands to the hardware for execution. Early experiments with DRC-Hubo showed that executing in open loop is problematic. Due to stiff closed kinematic chains, small errors in contact can cause excessive contradictory torques on the joints, leading to motor damage and/or shutdown. We employ a compliance controller to remedy this problem through current-based torque control. Because DRC-Hubo already has passive compliance in its ankles we found the compliance is only needed on the arms; as the robot closes its hand into a secure grip, it corrects errors on the hand position relative to the railings.

III. Ladder Climbing Motion Planning

The motion primitive planning approach is a hybrid of gait-based and sampling-based planners that blends the speed of gait-based techniques with the generality of sample-based ones [5]. Early work on motion planning for humanoid robots have studied gait-based methods for footstep planning [9], integration of footstep planning and vision [10], and selection of locomotion styles/postures to pass through narrow spaces [11]. These are most applicable to flat terrain with obstacles, and have limited applicability to highly variable terrain. More recent sampling-based techniques have allowed the robot to generate motions on highly varied terrain by reasoning directly with motion constraints [12]. Since ladders and stairs are semi-structured, it is reasonable to believe that *a small number of predetermined gaits can be applied to new ladders with a small amount of adaptation if the planner is sufficiently powerful*. Experiments in this paper confirm this intuition.

A. Definitions of Motion Constraints

Balance, actuator constraints, and collision avoidance are critical during climbing. The robot begins and ends on a two foot stance, and during climbing, it makes contact with three or four limbs. To model this, we introduce the concept of a *hold* to represent the overall contact of a single limb with the environment. The feasibility of plan depends on both the robot's configurations and contact state during the motion.

A hold includes multiple point-contacts $r_1, \ldots, r_k$ on the robot limb meeting $x_1, \ldots, x_k$ on the environment, and their contact normals $n_1, \ldots, n_k$ and friction coefficients are used to model the range of forces applicable to the environment (Fig. 6). In our implementation, we model a hand hold (with fingers) with eight point-contacts when the hand makes a tight grasp on the rail (see Fig. 6 (a)); and the foot hold with vertically-oriented point-contacts (see Fig. 6 (b)) at the corners of the contact region (four on a rectangular rung and two on a cylindrical rung). A set of holds yields a *stance* σ that describes a contact state of the robot (Fig. 6 (c)).

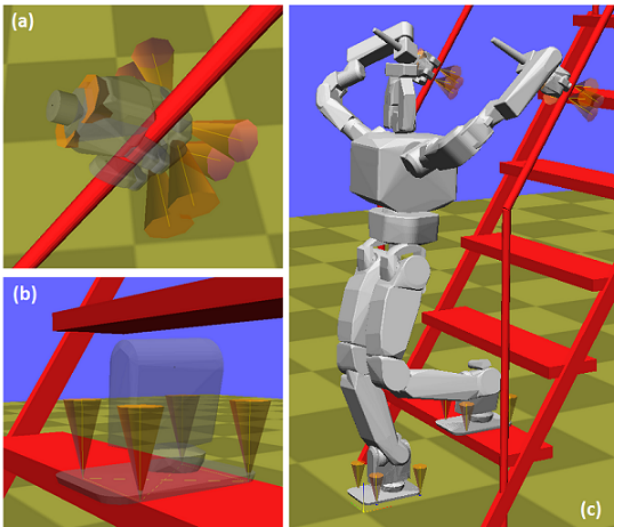

Fig. 6. (a) Hand hold modeling (b) Foot hold modeling. Each hold includes a set of point-contacts, the contact normals and the friction cones. (c) A configuration at the stance with two hands and two feet holds. Friction coefficient is assumed to be 0.25 in our example.

A configuration q at σ is defined as *feasible* iff it satisfies all the following constraints:

1) Contact constraints: robot maintains the point-contacts specified by all the holds in σ. This constraint restricts the motion to a lower-dimensional submanifold of configuration space.
2) Joint limit constraints: $q \in [q_{min}, q_{max}]$.
3) Collision free with the robot itself and the environment.
4) Quasi-static equilibrium: gravity balances the contact forces respecting friction, torque and grip force limits.

A linear program is formulated to check if there exists contact forces $(f_1, \ldots, f_k)$ respecting force limits at the given configuration q at stance σ for achieving balance (a method similar to [13]), where $f_1, \ldots, f_k$ are the contact forces for all the point-contacts $r_1, \ldots, r_k$ in σ respectively.

We define the feasible set of configurations at a stance as $F_\sigma = \{q \mid q \text{ is feasible at } \sigma\}$.

The planner outputs a continuous path of configurations $q(t)$ and stances $\sigma(t)$ such that the configuration is always feasible at its given stance. Stance switches are assumed to occur at discrete points in time.

B. Motion Primitives

The planner accepts two types of motion primitives, with each motion primitive corresponding to a specific sub-task:

1) Limb transfer. Such primitives invoke one hold change: add or remove the contacts of one limb while keeping other holds unchanged. E.g., placing the hands on the ladder from the standing pose will create new contacts on the hands while keeping the contacts on the feet unchanged; moving the left foot onto a higher rung will change its contacts to higher rung while keeping the rest holds unchanged.
2) Center of mass (COM) transfer. These primitives transfer the COM position without changing the stance. E.g., before moving left foot onto a higher rung, the COM is transferred to the right foot to achieve balance. In our implementation, there are three motion primitives for adjusting COM position, transfer COM to left/right foot before moving one foot, and transfer COM to the center of the two feet before releasing hand contacts.

Our implementation composes a ladder-climbing motion out of two motion primitive sequences, for a total of 9 primitives (see Fig. 7). The first sequence (primitives 1–2 in Fig. 7) mounts the ladder. The second executes a single climbing cycle of 7 primitives that ascends one rung (3-9 in Fig. 7). Furthermore, motion primitives for dismounting are designed specially for different ladders according to the top platform.

C. Motion Primitive Adaptation Planner

COM transfer tasks are relatively simple to plan, and simply require checking for the existence of a linear COM trajectory to the support foot. Limb transfer tasks are more complex because they require finding new contact positions as well as configuration-space paths. Here we describe the primitive-guided sampling-based planner in detail.

First, the planner samples hand and foot placements for limb transfer primitives, incorporating the human designer's knowledge from the motion primitives. Each such primitive defines a seed hold representing the "ideal" set of point-contacts relative to the ladder model and the robot's current configuration. Given the starting stance σ_s, the ending stance σ_e of current sub-task is generated by sampling a hold h_d around the seed hold for the moving limb while keeping the rest holds unchanged. The transition stance is $\sigma_t = \sigma_e - h_d$, which excludes the hold for the moving limb.

Second, the planner uses the discretized configurations of the motion primitive $\bar{Q} = (\bar{q}_s, \bar{q}_1, \ldots, \bar{q}_k, \bar{q}_e)$ to generate a feasible configuration sequence $Q = (q_s, q_1, \ldots, q_m, q_e)$ that satisfy:

1) $q_s \in F_{\sigma_s}$
2) $q_1, ..., q_m \in F_{\sigma_t}$
3) $q_e \in F_{\sigma_e}$

To do so, it calculates an ending configuration q_e by projecting $\bar{q}_e$ on to the contact constraints of F_{σ_e} using a numerical inverse kinematics (IK) solver. We use a Newton-Raphson technique with random restarts from increasingly perturbed initial configuration $\bar{q}_e$. Perturbations help increase the success rate by avoiding local minima for hard problems, while staying close to the seed $\bar{q}_e$. With each successful projection, collision is checked, and configurations that collide with the environment are retracted by solving a nonlinear constrained optimization process, similar to the Iterative Constraint Enforcement algorithm [13]. Algorithm 1 describes the projection procedure. The iteration limit n is set to be 10 in our implementation.

Algorithm 1 Project a configuration $\bar{q}$ onto F_σ

```
for i = 0, 1, ..., n: do
    q ← Solve-IK(q̄ + perturb(i))
    if IK was successful: then
        if q is collision free and stable: then
            return q
        if q has no self-collisions: then
            if retract(q) succeeds and q is stable: then
                return q
```

Finally, the intermediate configurations $(\bar{q}_1, \ldots, \bar{q}_k)$ of $\bar{Q}$ encode the waypoints for avoiding obstacles. Because they are designed for a certain ladder, they need adjustment to the new problem. We begin by performing a linear transformation that transforms $\bar{Q}$ to begin and end at the new (q_s, q_e) using a method similar to [5] (see Fig. 5). Again, perturbations are used to retract configurations that collide with the environment. Once milestones $(q_s, q_1, \ldots, q_k, q_e)$ are obtained, a recursive Bezier projection technique is used to smoothly interpolate between milestones while maintaining the contact constraints of F_{σ_t} [14]. A final feasibility check is then performed on the path.

The Motion Primitive Adaptation Planner is given by the following pseudocode:

Adapt-Primitive($\bar{Q}$, q_s, σ_s)

1. Sample ending stance σ_e from σ_s
2. Calculate $q_e \in F_{\sigma_e}$ (Algorithm 1)
3. Linear transform $\bar{Q}$ according to (q_s, q_e)
4. Utilize $\bar{Q}$ for generating milestone path Q
5. Fine discretize Q and check feasibility

If any step fails, the algorithm returns failure; otherwise, it returns a feasible path for the sub-task.

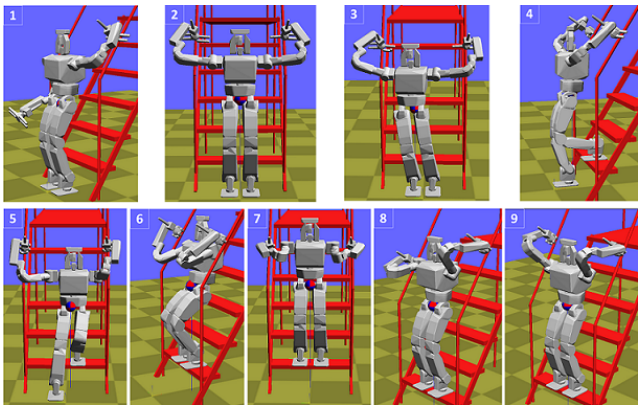

Fig. 7. Motion primitive examples, with the final configuration of each primitive shown. From left to right, top to bottom: mountLH, mountRH, moveCOMtoRF, placeLF, moveCOMtoLF, placeRF, moveCOMtoCenter, placeLH and placeRH where L and R indicate left and right, H and F indicate hand and foot respectively.

D. Multi-Step Planning

To compose primitives together into an N-primitive climbing sequence, the planner performs a depth first search. This approach allows the possibility of backtracking to revise a prior choice if the current move appears infeasible or hard. The recursive multi-step planning procedure is given by the following pseudocode:

MSP-Recurse(q_i, σ_i)
1. Get the next primitive $\bar{Q}_{i+1}$
2. For $j = 0, \ldots, m$:
3. If Adapt-Primitive($\bar{Q}_{i+1}$,q_i,σ_i) succeeds:
4. Let q_{i+1}, σ_{i+1} be the sampled endpoint.
5. If $i + 1 = N$, return the path leading to q_{i+1}, σ_{i+1}.
6. If MSP-Recurse(q_{i+1},σ_{i+1}) succeeds, return the path.
7. Return failure

Since the motion planner sequentially plans each sub-task, the feasibility of one sub-task is affected by the ending configuration and stance of the last sub-task, which may have depended on the one before that, etc. The parameter m controls the amount of effort the planner gives to each sub-task before backtracking. At one extreme, $m = 1$ implements a descent approach with random restarts, which gives up too quickly; at the other, the planner may get stuck with no progress. In our tuning, we found the value $m = 5$ to have good performance in general.

IV. Trajectory Execution via Compliance Control

Execution of motion on a physical robot is never perfect. Errors from calibration, modeling, control, and mechanical slack result in the robot's end-effectors being slightly misplaced. Furthermore, there are always three or four limbs in contact with the environment which form multiple closed kinematic chains. In joint position control mode, DRC-Hubo tends to use motors controllers with extremely high position gains, but these high gains tend to result in an over-current whenever a closed kinematic chain is present, due to the aforementioned small errors. When a motor detects an over-current condition, the joint motor controller cuts off power to the motor to prevent damage to itself and the system as a whole.

To solve this, we implement a passive compliance controller, which controls the pulse width modulation (PWM) directly on the motors to adjust output torque according to a compliant set of position gains. The PWM duty percentage P is computed using eq. (1), where k_P and k_D are proportional and differential gains; x_d, x_c, $\dot{x}_c$ are the desired joint position, current joint position and velocity respectively; f is the feedforward term which is intended for investigating gravity compensation in future work.

$$P = k_P \cdot (x_d - x_c) + k_D \cdot (-\dot{x}_c) + f \tag{1}$$

In the experiments, we keep k_D and f as zeros and rely on the internal friction to damp the motors. k_P is used to adjust the joint compliance.

We enabled compliance on the arm joints which are mainly used for extra balance support. Leg joints are kept stiff to support the weight of the body. Passive compliance control on the arms makes them compliant to tolerate errors. However, without gravity compensation, compliant joints also introduce errors in the execution. We mitigate the execution errors by tuning the compliance gains to control the level of compliance. Forward kinematics implies that shoulder, elbow and wrist joint positions have decreasing effects on the end-effector pose, so we set the compliance gains k_P for arm joints to achieve an increasing level of compliance from shoulder to wrist.

The compliance control is implemented on Hubo-ach, a low level controller based on the ACH Inter-Process Communication (IPC) library [15]–[17] developed specially for Hubo robots.

V. Experiments

This section tests the performance of our proposed system (see the video supplement). For all timing results, the planning was carried out on an Intel Core i7 2.8 GHz machine with 4GB RAM. Simulation results use a custom simulator based on the Open Dynamics Engine rigid body simulator, extended with robot sensor / controller simulation and improved collision handling.

A. Planning on a Variety of Ladders

Our motion planner has been tested with the DRC-Hubo robot, as well as its earlier version Hubo-II+, on various ladders using computer simulations. Fig. 8 shows the robot climbing three different ladders. The motions are generated within approximately 10–15 s.

We also tested the capabilities of DRC-Hubo to climb a range of ladders while varying two parameters: rung spacing in the range [20 cm,35 cm], and incline angle in the range $[70°, 90°]$. The motion primitive in this experiment is designed specifically for $80°$ with 25 cm inclination. For each ladder, we tested our motion planner by utilizing motion primitives to generate motions for 2-step climbing (see Fig. 7 for the utilized motion primitives). The planner was given a 120 seconds cutoff planning time. Among all the 336 ladders,

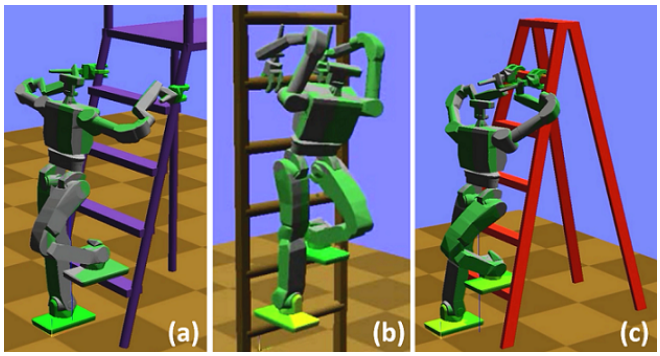

Fig. 8. DRC-Hubo climbs different ladders of 30cm rung spacing in physical simulation. The green shadow indicates the reference motion. (a) Climbing a regular ladder of 70-degree inclination. (b) Climbing a vertical ladder. (c) Climbing an A-frame ladder of 60-degree inclination.

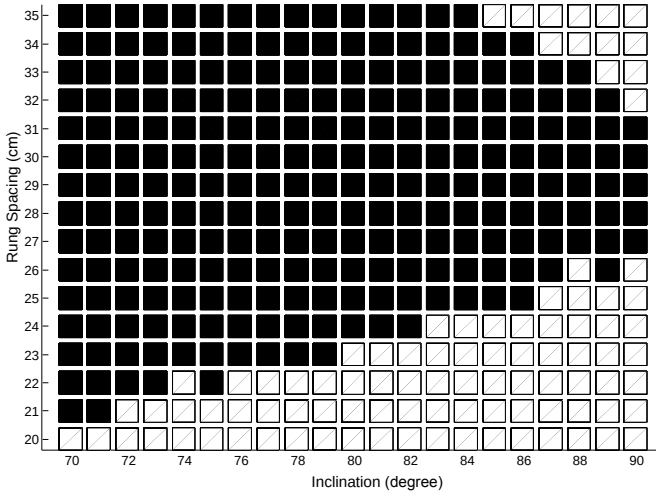

Fig. 9. Testing on a range of ladder inclinations and rung spacings in simulation. Black indicates motion planner succeeded in planning the two-step climbing motion within 120 s cutoff time.

72% of them can be solved successfully (see Fig. 9). Due to the geometry of the foot and ankle, the planner failed at rung spacings lower than 22 cm because the rung spacing is not high enough to avoid collision between the ankle and the higher rung. When the inclination is close to 90° and rung spacing is close to 35 cm, the possible reason of failure is the set of motion primitives may not be easy to be adapted to such ladders.

B. Motion Robustness and Error Toleration

We performed three robustness tests. In the first test, we tested the amount of modeling error tolerable by the hand on the real robot. Experiments in open-loop mode showed frequent arm motor shutdown due to excessive torque caused by control and calibration errors. With the compliant control, we perturbed the hand position increasingly such that it deviates from the desired position for gripping the rail. We found that the hand pose can be adjusted to grip the rail as long as the rail is within the grasping range of fingers.

Second, we tested in simulation whether the robot can continue the climbing motion when the rung spacing is higher or lower than the planned value. On a ship ladder example, we increase (reduce) the rung spacing 1 cm each time and run the planned trajectory in the changed scenario. The results show that the robot is able to tolerate such errors with up 2 cm variation from the planned rung spacing. At 3 cm, the robot's foot slips from its intended placement on the rung and the robot falls over.

In the final simulation test, we dropped an object on the robot during execution (see Fig. 10). Such disturbances might occur due to falling debris in disaster environments or under trees. We gradually increased the weight of the object by 5 kg increments, and the robot is able to withstand a 10kg impact at speed 5.5m/s. At 15 kg, the robot is unable to hold onto the ladder and it falls.

C. Climbing a Ship Ladder on Physical Robot

Fig.11 demonstrates our system applied to a standard industrial ship ladder that is similar to the one in DARPA's specifications "DRC Trials Initial Task Descriptions" for the DRC Trial [8]. The ladder has 5 rungs and a wide top

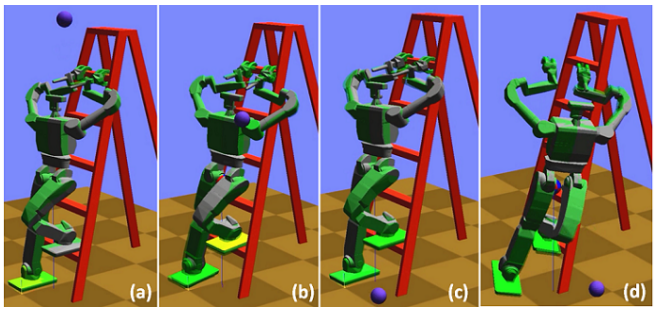

Fig. 10. Robustness test (a), (b) A ball of progressively increasing weight is dropped onto the robot at the speed of 5.5m/s. (c) The robot is able to continue the motion after hit by the ball of 10 kg. (d) The robot falls down after hit by the ball of 15 kg.

for dismounting. It is 60° inclination, 80 cm rung width, 17 cm rung depth, 25 cm rung spacing and 100 cm rail height. Planning completed in less than 15 seconds, and the climbing motion successfully completes in 7 minutes. The robot is also able to climb back down the ladder simply by reversing the motion (not shown).

VI. Conclusion and Future Work

This paper presented a planning and control framework for ladder climbing humanoid that can be successfully applied to a wide variety of ladders. The method integrates a motion primitive-based motion planner with a compliant controller, and the resulting system is demonstrated to enable DRC Hubo to successfully climb an industrial ladder.

Several issues remain for ongoing work. We are currently integrating automated and human-assisted perception into the system via deformable model fitting, and we are investigating methods for detecting and correcting for larger execution errors using visual or tactile feedback. Climbing is also somewhat slow because quasistatically-stable paths must be executed slowly to avoid dynamic effects, so we are investigating methods (e.g., [14]) to optimize execution times under

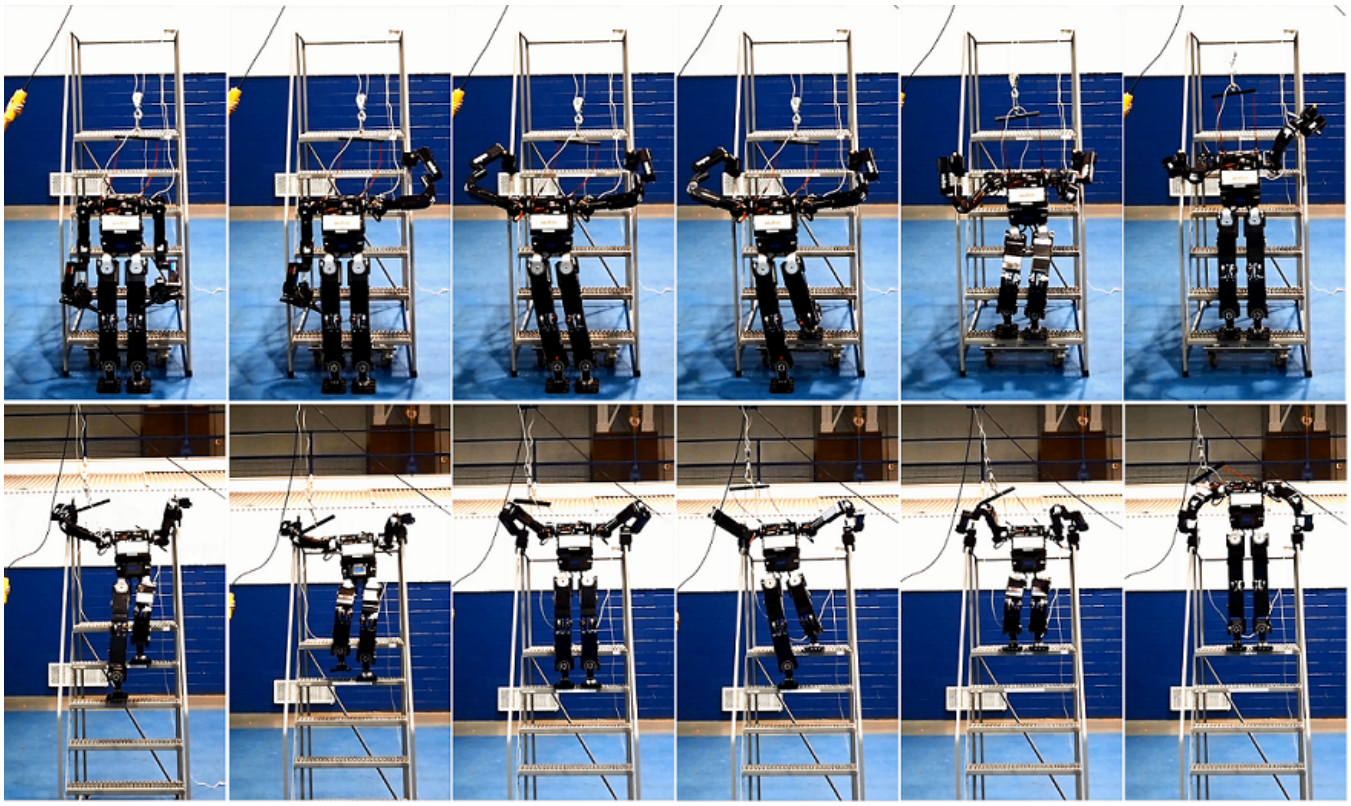

Fig. 11. Snapshots of DRC-Hubo mounting, climbing and dismounting the ship ladder.

dynamic constraints. Finally, we would like to improve the versatility of the planner to adapt to very different strategies and scenarios. As an instance of the latter issue, note the planner failed to solve for a vertical ladder with 35 cm rung spacing. We cannot tell whether the robot is physically incapable of climbing this ladder, or the primitive was not easy enough to adapt. Future implementations may use large primitives libraries and dynamically select appropriate ones for the given environment.

REFERENCES

[1] H. lida, H. Hozumi, and R. Nakayama, "Development of Ladder Climbing Robot LCR-1," *J. Robotics and Mechatronics*, vol. 1, no. 4, pp. 311–316, 1989.

[2] D. Bevly, S. Farritor, and S. Dubowsky, "Action module planning and its application to an experimental climbing robot," in *IEEE Int. Conf. Robot. Autom.*, vol. 4, 2000, pp. 4009–4014 vol.4.

[3] S. Kim, M. Spenko, S. Trujillo, B. Heyneman, V. Mattoli, and M. Cutkosky, "Whole body adhesion: hierarchical, directional and distributed control of adhesive forces for a climbing robot," in *IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 1268–1273.

[4] H. Yoneda, K. Sekiyama, Y. Hasegawa, and T. Fukuda, "Vertical ladder climbing motion with posture control for multi-locomotion robot," in *IEEE/RSJ Int. Conf. Intell. Robot. Sys.*, Sept. 2008, pp. 3579–3584.

[5] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots," in *Workshop on the Algorithmic Foundations of Robotics*, 2006.

[6] T. Bretl, S. Lall, J.-C. Latombe, and S. Rock, "Multi-step motion planning for free-climbing robots," in *Workshop on the Algorithmic Foundations of Robotics*, 2004.

[7] T. Bretl, "Multi-step motion planning: Application to free-climbing robots," *PhD Thesis, Stanford University*, June 2005.

[8] (2013) The darpa robotics challenge website. [Online]. Available: http://www.theroboticschallenge.org

[9] J. J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-stable motion planning for humanoid robots," *Autonomous Robots*, vol. 12, pp. 105–118, 2002.

[10] S. Kagami, K. Nishiwaki, J. Kuffner, K. Okada, M. Inaba, and H. Inoue, "Vision-based 2.5D terrain modeling for humanoid locomotion," in *IEEE Int. Conf. Robot. Autom.*, vol. 2, Sept. 2003, pp. 2141–2146.

[11] F. Kanehiro, H. Hirukawa, K. Kaneko, S. Kajita, K. Fujiwara, K. Harada, and K. Yokoi, "Locomotion planning of humanoid robots to pass through narrow spaces," in *IEEE Int. Conf. Robot. Autom.*, vol. 1, 2004, pp. 604–609.

[12] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *Robotics: Science and Systems*, vol. 27, no. 11-12, pp. 1325–1349, Dec. 2008.

[13] K. Hauser, T. Bretl, and J.-C. Latombe, "Non-gaited humanoid locomotion planning," in *IEEE Int. Conf. on Humanoid Robots*, Dec. 2005, pp. 7–12.

[14] K. Hauser, "Fast interpolation and time-optimization on implicit contact submanifolds," in *Robotics: Science and Systems*, Berlin, Germany, June 2013.

[15] N. Dantam and M. Stilman, "Robust and efficient communication for real-time multi-process robot software," in *IEEE Int. Conf. on Humanoid Robots*, 2012, pp. 316–322.

[16] M. Grey, N. Dantam, D. Lofaro, A. Bobick, M. Egerstedt, P. Oh, and M. Stilman, "Multi-process control software for hubo2 plus robot," in *Proc. Int. Conf. on Technologies for Practical Robot Applications (TePRA)*, 2013, pp. 1–6.

[17] D. Lofaro, "Unified algorithmic framework for high degree of freedom complex systems and humanoid robots," Ph.D. dissertation, Drexel University, College of Engineering, Electrical and Computer Engineering Department, May 2013.