Planning Complex Physical Tasks for Disaster Response with a Humanoid Robot

Hao Dang, Youngbum Jun, Paul Oh, and Peter K. Allen

Abstract— **Humanoid robots are attempting ever more complex tasks in lieu of humans. Disaster response is a promising area for the use of humanoids due to safety concerns. However, controlling a high DOF humanoid robot to autonomously perform a complex task in unknown and unstructured environments is challenging. In this paper we describe a simulation framework for humanoid grasping and transport tasks that includes dynamics, and is easily ported to a real physical humanoid robot. The system can be used to rapidly prototype humanoid motions and dynamics in simulation, and can then be ported to the physical hardware. Experimental results are presented, both in simulation and physical experiments, with the HUBO humanoid, on a task from the DARPA Robotics Challenge, attaching a fire hose to a hydrant.**

I. Introduction

Over the last several years, a new generation of humanoid robots has emerged that show great promise in being able to accomplish complex tasks associated with human behavior [1], [2], [3], [4], [5]. New algorithms have been developed to improve robotic grasping [6], [7], [8] and manipulation [9], [10], [11] capabilities. A recent push in this field has been the DARPA Robotics Challenge [12]. DARPA has set up a scenario for large-scale disaster remediation, based loosely on the Fukushima Daiichi Tsunami event in 2011. The scope of this disaster was very large, and the hazardous conditions caused by the radioactive contamination made it extremely difficult for a timely and safe human response to the disaster. Given the hazards involved, and the quick response needed, humanoid robots seem to be a promising technology in mitigating future events in a safe and timely manner.

DARPA has identified a number of tasks that a humanoid robot might possibly accomplish in such a scenario. They include:

1) Drive a utility vehicle at the site.
2) Travel dismounted across rubble.
3) Remove debris blocking an entryway.
4) Open a door and enter a building.
5) Climb an industrial ladder and traverse an industrial walkway.
6) Use a tool to break through a concrete panel.
7) Locate and close a valve near a leaking pipe.
8) Replace a component such as a cooling pump.
9) Attach fire hose couplings.

It is clear that there are many other possible task scenarios related to such a widespread disaster event, and even with planning and foresight, new and more complex tasks may be required of any humanoid remediation effort. To develop humanoid capabilities in this area, we have begun to build a hybrid simulation system that can be used to simulate specific tasks, and which can be used to ascertain if the humanoid is capable of the task. This environment can also be used to synthesize new capabilities and designs for a humanoid that can then be translated into working physical hardware.

In this paper we present our work on the hose attachment task. We first discuss the task specification and the approach we take to accomplish the task. We explain in detail our manipulation pipeline which includes grasp recording, motion planning, and motion execution. We also show our initial experiments in both simulation and physical settings using a real physical humanoid robot, the HUBO II plus [13].

II. Task Specification

A. Overview

In general, three fundamental processes are required for a humanoid robot to complete a task autonomously. First, the humanoid robot has to understand its surrounding environment. Second, the robot needs to plan the motion appropriate to the given environmental and task information. Third, the robot needs to control itself in order to physically perform the desired motion and task. Based on this model, we divide our work on the hose installment task into three parts - visual and spatial recognition, kinematic motion planning, and dynamic motion control. In the following, we will discuss these three parts. However, as initial work, we focus in this paper on our work in kinematic motion planning and porting planned motion and grasps onto a physical real robot.

Visual and spatial recognition - The main objective is to provide the accurate environmental information. The information is used to reconstruct the workspace of the robot for motion planning. Currently we use a motion capture system as our main tool, but we will be utilizing a combination of stereo vision and radar for accurate sensing in the future.

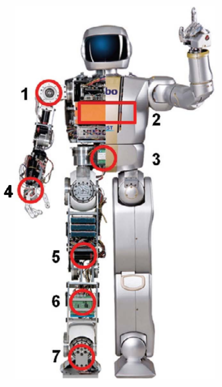
Kinematic motion planning - Based on the visual and spatial information about environment, joint trajectories are planned and evaluated during a motion planning process. These joint trajectories define a collision-free paths to grasping and manipulating the target object.

Dynamic motion control - This part is critical for the physical performance of the robot. The ultimate goal is to

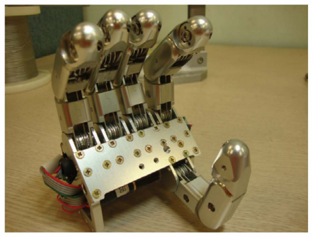
This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) award #N65236-12-1-1005 for the DARPA Robotics Challenge

Hao Dang and Peter K. Allen are with Department of Computer Science, Columbia University, {dang,allen}@cs.columbia.edu

Youngbum Jun and Paul Oh are with Department of Mechanical Engineering, Drexel University, yj55@drexel.edu paul@coe.drexel.edu



(a) The HUBO Robot



(b) The HUBO Hand

Fig. 1. The HUBO robot and the robot hand. In Figure 1(a): 1. Harmonic drive; 2. Two computers and batteries placed in the chest; 3. Inertial measurement unit; 4. F/T sensor on the wrist; 5. Brushless DC motor; 6. Motor control board; 7. F/T sensor on the ankle.

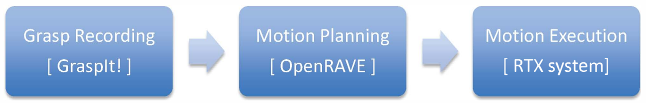


Fig. 2. Our manipulation pipeline which includes three stages. In the grasp recording stage, stable grasps are predefined and stored into a grasp database using our *GraspIt!* simulator [14]. In the motion planning stage, a predefined stable grasp is retrieved from the database according to the requested manipulation tasks and arm motion is planned using the OpenRAVE simulator [15] to generate a joint trajectory for the manipulation task. Once a joint trajectory is generated, it is sent to the robot through a real-time control system to accomplish the manipulation task.

minimize and compensate for the error caused by systematic and environmental uncertainties.

B. Our Approach

The hose installment task requires the integration of different planning and control algorithms. We decompose the task into the following four sub-tasks and describe our approach to each of them.

The first sub-task for the humanoid robot is to detect where the hose is and to plan a collision-free motion to grasp it. In our work, the fire hose is assumed to be rigid. We are planning to put force/torque sensors on the robot hand so that contact forces can be measured by them. With this sensory capability, a force compensator program can be used to control the grasping motion. In addition, the grasping force controller utilizes the tactile sensors on the robot hands and controls the net force of holding, which will optimally avoid slipping and assure stable contact with the hose.

After the hose is grasped, the humanoid robot lifts up the hose so that there can be some clearance between the hose and the table to facilitate motion planning for transporting the hose. In this process, the robot may need to adjust its body pose properly with a center of mass position (CoM) controller to keep the zero-moment point (ZMP) of the robot within the support polygon.

The third sub-task is to carry the hose to the target position. To this end, the humanoid robot first detects the target and obstacles. Then it generates a collision-free path that defines the walking path and the foot placement in real time. Considering the change to the location of the original CoM of the robot caused by the extra weight of the hose, the walking gait may be adjusted based on the new CoM position. In the meantime, the ZMP controller keeps controlling the robot to follow the desired ZMP trajectory defined by the walking path and foot placement. A separate landing controller compensates for the unexpected contact forces caused by the ground contact.

Once the hose is transported to the hydrant, the humanoid robot moves the hose to the hydrant before it inserts the hose into the hydrant. Considering potential errors from the kinematic model of the robot and its sensors, the humanoid robot may utilize a dedicated force controller to compensate for unexpected contact force and to predict whether the hose is properly oriented and aligned with the hydrant.

In this paper, our focus is on planning arm trajectories for all the motions to grasp the hose, transport it to the hydrant, and insert the hose into the hydrant. While we haven't implemented all the dedicated controllers as specified above, we have ported the planned motion to the physical robot and executed the motions as our initial result.

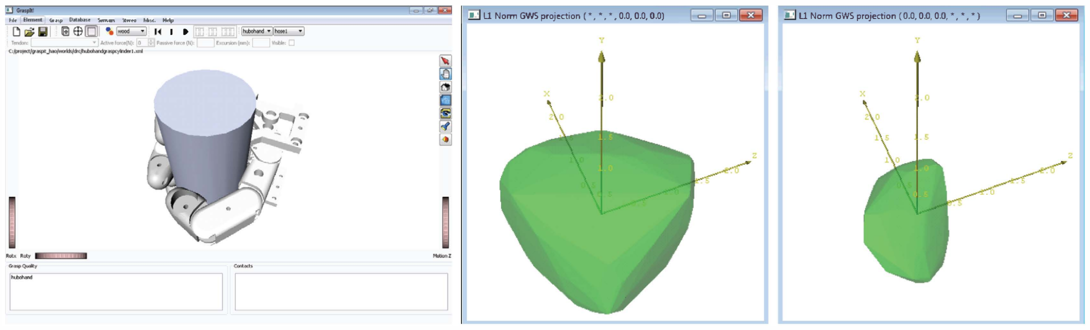
III. The HUBO Robot

A. Hardware Overview

HUBO is a full-sized humanoid robot and open-platform developed at KAIST (Figure 1(a)). It is 130cm tall and weighs about 42kg. In terms of mechanics, HUBO has 38 degrees of freedom including 6 on each leg, 6 on each arm, 5 on each hand, 1 on the torso, and 3 on the neck. Each joint is driven by a brushless DC motor with harmonic drive and has an optical encoder that allows users to control each joint angle individually. There are four 3-axis force-torque sensors located at each wrist and ankle joint, and one inertial measurement unit at the center of pelvis. Inside the robot, two computers are located in the chest. One is dedicated to joint control and sensor measurement via CAN bus and the other one works for other processes such as perception and high-level control. The real-time operating system (RTX) and customized control boards are the main features of the HUBO robot which compensate for the delay and interpolate the error so that a user can control each actuator at a hardware-based frequency [16].

There are two robot hands on the HUBO robot. In each of the HUBO hands, as shown in Figure 1(b), there are five fingers connected through cables. Each of these fingers is driven by a small DC motor. The net grasping force is 15 N.

We use the OpenRAVE software as a simulation environment to simulate the HUBO robot and plan robot motions [15]. This simulator provides us with a tool to rapidly prototype and evaluate different algorithms and analyze the motion of the HUBO robot. One main feature of OpenRAVE is the open-simulation module that is kinematically and dynamically synchronized with the HUBO robot, which allows users to apply their algorithms to the physical robot and evaluate the results on the HUBO robot. Figure 5


L1 Norm GWS projection (*, *, *, 0.0, 0.0, 0.0)
L1 Norm GWS projection (0.0, 0.0, 0.0, *, *, *)

(a) (b) (c)

Fig. 3. Analyzing a robotic grasp using the *GraspIt!* simulator. Figure 3(a) shows a HUBO hand grasping a cylinder inside the *GraspIt!* environment. Figure 3(b) shows the grasp wrench space projected onto the 3-dimensional force space $[f_x, f_y, f_z]$. Figure 3(c) shows the grasp wrench space projected onto the 3-dimensional torque space $[t_x, t_y, t_z]$.

shows a simulated HUBO robot loaded in the OpenRAVE environment using the OpenHubo package which has been developed by Drexel University. This package contains a full dynamic model of the HUBO robot and a simulated servo controller to simulate robot motions in dynamics.

IV. Manipulation Pipeline

Our manipulation pipeline consists of three stages: grasp recording, motion planning, and motion execution. Figure 2 illustrates the logic components of the manipulation pipeline we followed in our work. At the grasp recording stage, we predefined stable grasps on different objects using our robotic grasp simulator, *GraspIt!* [14]. At the motion planning stage, a stable grasp was retrieved from the grasp database according to the requested manipulation tasks. Joint trajectories are then generated which will move the robot arm to the grasping pose and accomplish the manipulation task. For the purpose of motion planning, we use the OpenRAVE simulator as our planning tool [15]. Once joint trajectories are generated, they are sent to a robot controller which executes these trajectories in the robot's workspace to finish the manipulation task. In this section, we discuss each stage in detail.

A. The GraspIt! Simulator

The *GraspIt!* simulator is a simulation tool which allows a user to create and analyze grasps of a given 3D object model with a given articulated hand model. Grasps can be performed either automatically, where the system closes the fingers around the object at preset velocities, or manually through direct manipulation of the joints. To evaluate the stability of a robotic grasp, this simulator provides two numerical measurements, the epsilon quality and the volume quality [17]. Both of these two measurements are based on the analysis of the convex hull of the grasp wrench space (GWS).

A GWS is a 6-dimensional space which contains a set of possible resultant wrenches produced by the fingers on the object. A wrench is a 6-dimensional vector $[f^{1\times3}, \tau^{1\times3}]$ that describes the combination of the possible force and torque. In our work, a GWS is generated by assuming the sum of the normal forces applied at each contact is 1. This assumption approximates a limited power source for the hand [18]. In geometry, the volume quality measures the volume of the potential wrench space and the epsilon quality measures the radius of the largest ball centered at the origin of a GWS and fully contained in the GWS.

The epsilon quality, ϵ, refers to the minimum relative magnitude of the outside disturbances that could destroy the grasp. So, when we take into account the limit of the maximum forces a robotic hand can apply, a grasp would be less stable if it has a smaller epsilon quality. This is because the smaller epsilon quality indicates that a relatively smaller outside disturbance can break this grasp even when the robotic hand has already applied the maximum forces it supports.

The volume quality, v, measures the volume of the potential wrench space generated by the grasp given unit contact normal forces. A grasp with a larger potential wrench space would require less forces at each contact than grasps with smaller potential wrench spaces. This indicates that the larger the volume quality is, the stronger the grasp could be.

As an example, Figure 3(a) shows a snapshot of the *GraspIt!* simulator analyzing a HUBO hand grasping a cylindrical pipe. Figure 3(b) shows the GWS projected to the 3-dimensional force space $[f_x, f_y, f_z]$ and Figure 3(c) shows the GWS projected to the 3-dimensional torque space $[t_x, t_y, t_z]$. Because the origin of the wrench space is contained in the GWS, this grasp is force-closure.

B. Grasp Recording

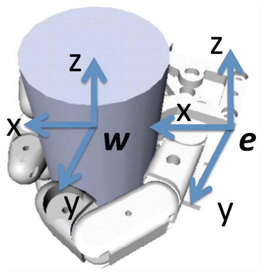
The *GraspIt!* simulator allows us to manually define grasps on different objects and analyze their stability. In our work, we use this tool to define and record stable grasps of a HUBO hand on objects to be manipulated. To do this, we first load the object model and the hand model into the *GraspIt!* simulator. We manually move the robot hand to the ideal wrist pose and close the fingers around the object. The fingers stops closing once the collision detection system in the *GraspIt!* simulator detects contacts between fingers and the object. Through this process, a grasp is defined. Using the stability analysis utility, we also evaluate the stability of this grasp. For stable grasps which have positive epsilon and volume qualities, we record the wrist pose of the robot hand and the joint angles of the fingers. These numbers can later be used to specify the hand posture to grasp the corresponding object. As an example, Figure 3(a) shows a stable grasp of a HUBO hand on a pipe which is obtained through this process.

C. Motion Planning

In the previous step, we have recorded stable grasps on different objects. We now need to plan joint trajectories to move the arm to the grasping pose, execute the predefined grasp, and accomplish the rest of the manipulation task. Generating joint trajectories can be achieved by using an existent motion planner. In our work, we exploit the motion planner proposed by Berenson et al. [19]. This motion planner is based on Constrained Bi-Directional Rapidly-exploring Random Tree (CBiRRT) and plans on constraint manifolds induced by pose constraints as well as other constraints. These constraints are user-defined and are very useful for object manipulation tasks. For example, we can use these constraints to plan a motion to transport a plate from one place to another while keeping it horizontal along the path. We can also use these constraints to specify grasping affordances for different objects. For example, since a cylindrical pipe is rotationally symmetric around its center, any grasps rotated around that axis would be equivalent and they shall not be differentiated. In this CBiRRT planner, we can specify this symmetry as a grasping affordance and allow other equivalent grasps to be considered.

To represent pose constraints and affordances, Task Space Regions (TSR) were introduced by Berenson et al. in their previous work [20]. In the following, we give a brief introduction to it. Interested readers please refer to their original work in [19] for more details and explanations. A TSR consists of three parts: 1) a transform T_w^0 from the origin of the world to the TSR frame w, 2) a transform T_e^w from the TSR frame w to the end effector e, and 3) a 6×2 matrix B of bounds in the coordinates of the TSR frame w. The ideal pose of the end effector with respect to the world is given by $T_e^0 = T_w^0 \cdot T_e^w$. The bounds in B specify the potential offset that can be allowed with respect to the TSR frame w. Thus, for a CBiRRT planner, it samples the end effector pose as $T_{sample}^0 = T_w^0 \cdot T_{sample}^w \cdot T_e^w$, where T_{sample}^w is generated by sampling the bounding space B.

Figure 4 shows an example of a HUBO hand grasping a cylindrical pipe. In this grasp, we want to allow any grasp that is rotated around the length direction of the pipe to be considered in the CBiRRT planner. Thus, we define the TSR frame w at the origin of the pipe. By specifying the matrix $B = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -pi, pi]$ as in Figure 4, we allow the hand to freely rotate around the z axis in the TSR frame w. The transform of the TSR frame w specifies the pose of the pipe with respect to the world coordinate system and needs to be determined in the workspace, either through a perception system or manual measurements.



$$T_e^w = \begin{bmatrix} 1 & 0 & 0 & -0.02 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = [0,0,0,0,0,0,0,0,0,0,-pi,pi]$$

Fig. 4. TSR of a grasp on a pipe using a HUBO hand. The TSR frame w coincides with the origin of the pipe with its z axis aligned with the length direction of the pipe. The transform T_e^w specifies the relative pose of the robot hand with respect to the pipe in the pipe's coordinate system w. The bounds B specifies the grasping affordances where we allow free rotation around the z axis in w. In this configuration, any grasp by rotating the hand around the z axis in frame w will be treated equally in the CBiRRT planner.

D. Motion Execution

The resulting trajectory of the CBiRRT planner specifies the joint angles at different points of time, which is also a collision-free trajectory for the robot arm to move the end effector to the target grasping pose. With an extra re-timing process to the generated joint trajectory, we analyze the trajectory and make sure both the velocity and acceleration of each joint are within its limit. In this step, we apply this joint trajectory to the arm controller to execute the trajectory on a robot.

In our work, the robot is connected to a RTX real-time system which updates the motor commands at a fixed frequency. The upper body (arms and neck joints) is controlled at 100Hz and the lower body (legs and torso joints) is operated at 200Hz. Thus, a generated trajectory for the robot arm contains the values of each joint every 0.01 second. To execute such a trajectory, the RTX system sends each joint sample as a command to each motor controller via CAN bus. Once each motor controller receives the joint angle value, it generates PWM pulses to drive the DC motor. The PID position controller on each motor controller controls the joint angle with the optical encoder.

V. Experimental Results

To evaluate our work, we have done initial experiments in both simulation and physical settings.

A. Experiment in Simulation

The first experiment is done in simulation. As is described in Section II-B, we decomposed this task into four key phases: grasping the hose, lifting up the hose, transporting the hose to the hydrant, and inserting the hose into the hydrant. Figure 5 illustrates the process of this experiment in simulation where a simulated hose installment task is performed by a simulated HUBO robot using the OpenRAVE simulator.

In the beginning, the robot is initialized to a predefined pose as shown in Figure 5(a). The poses of the table, the hose, the hydrant, and the robot are manually predefined in an OpenRAVE environment file. Using the CBiRRT planner inside OpenRAVE, the arm motion for reaching to the hose can be generated. Figure 5(b) and 5(c) shows the right arm

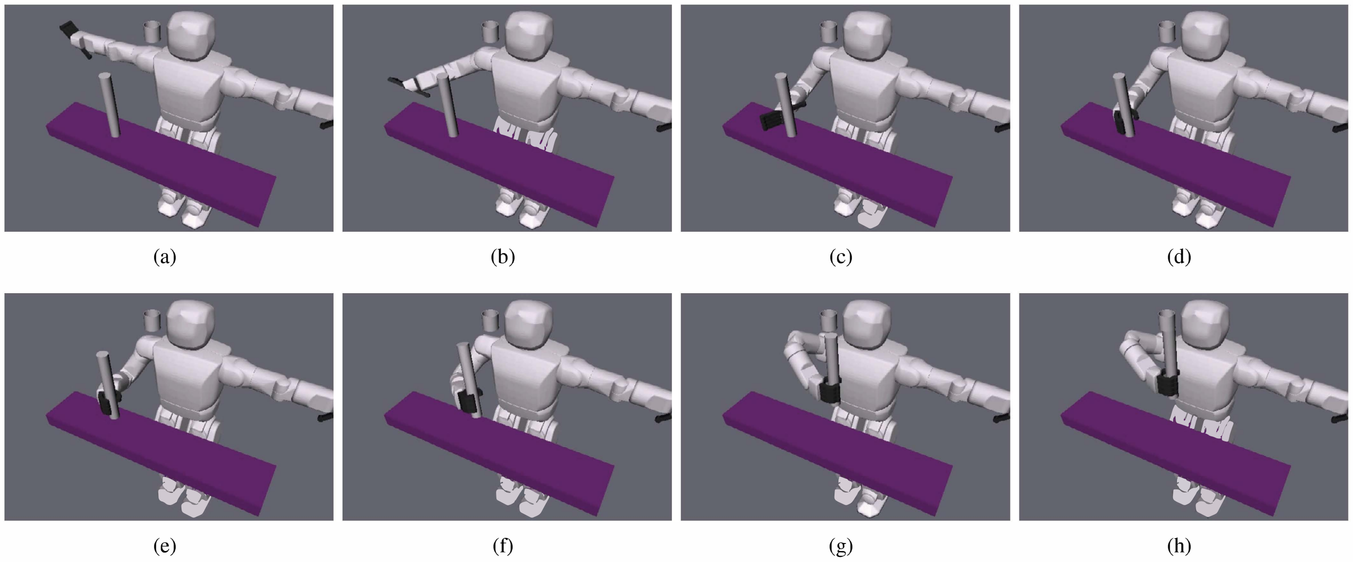


Fig. 5. Snapshots from a simulation experiment. Figure 5(a) shows the HUBO robot standing in its initial pose. Figure 5(b) and 5(c) shows the right arm reaching to the hose. Figure 5(d) shows the robot grasping the hose. Figure 5(e) shows the robot lifting up the hose from the table. Figure 5(f) and 5(g) shows the robot transporting the hose to the hydrant and inserting it into the hydrant as in Figure 5(h).

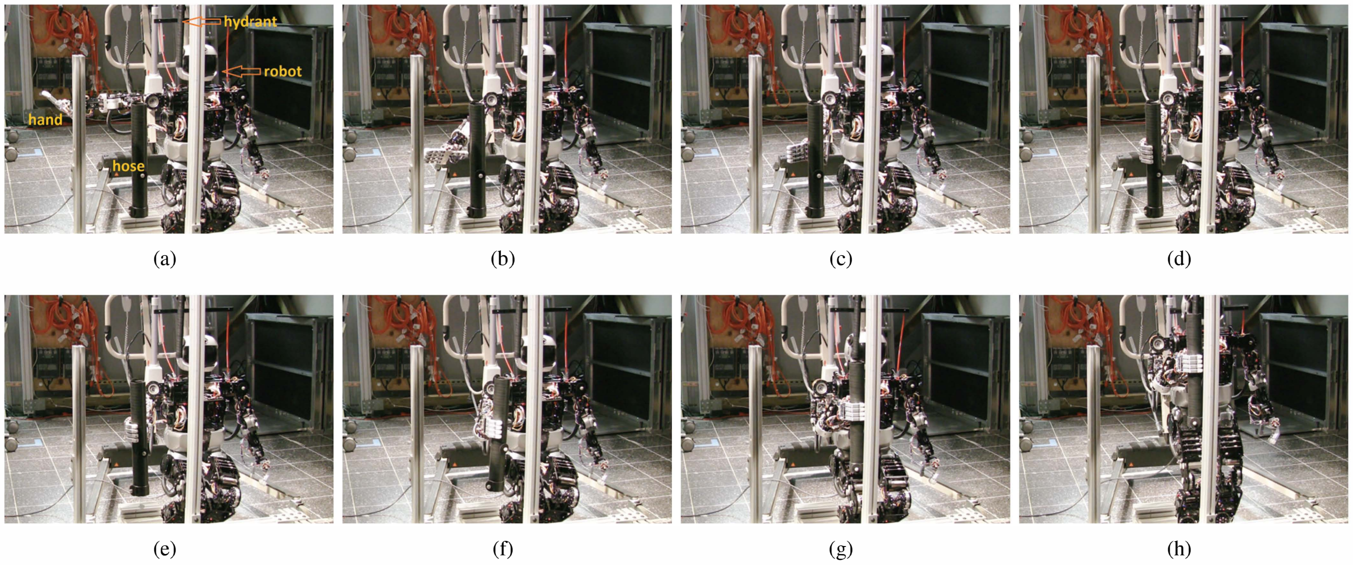


Fig. 6. Snapshots from a physical experiment with the HUBO robot performing a hose installment task with its right arm. The frames in front of the robot are part of the experimental setup. Figure 6(a) shows the HUBO robot standing in its initial pose. Figure 6(b) and 6(c) shows the right arm reaching to the hose. Figure 6(d) shows the robot grasping the hose by closing its fingers at the grasping pose. Figure 6(e) shows the robot lifting up the hose from the table. Figure 6(f) and 6(g) shows the robot transporting the hose to the hydrant and inserting it into the hydrant as in Figure 6(h).

moving from its initial pose and reaching the hose. Once the robot reaches the hose, it grasps the hose by closing the fingers as shown in Figure 5(d). Figure 5(e) shows the robot lifting up the hose from the table before it transports the hose to the hydrant, which is illustrated in Figure 5(f) and 5(g). Finally, when the hose is transported to the hydrant, the robot inserts the hose into the hydrant by moving the hose into the hydrant as shown in Figure 5(h).

B. *Experiment in Physical Settings*

In addition to experiments in simulation, we also performed experiments with a real physical robot, where we ported the motions planned in the OpenRAVE environment to a HUBO robot and executed the task with it. In the physical experiment, the locations of the hose and hydrant were defined using a motion capture system (optiTrack). Two markers were placed in the workspace to specify the locations of the objects. Once the motion capture system captured the locations of the markers used to specify the locations of the objects, we passed the locations to the OpenRAVE simulator to setup the environment for motion planning. With this simulation environment constructed, the three steps of our manipulation pipeline as we discussed in Section IV and Figure 2 took place to generate joint trajectories for the robot. Figure 6 illustrates the process of a HUBO robot performing the hose installment task in its workspace.

In the beginning, the robot is initialized to a predefined pose where the right arm is horizontal as shown in Figure 6(a). Following a trajectory generated by the CBiRRT planner inside OpenRAVE, the right arm moves from its initial pose to reach the hose as shown in Figure 6(b) and 6(c). Once at the grasping pose, the robot closes its fingers and grasps the hose as shown in Figure 6(d). Figure 6(e) shows the robot lifting up the hose from the table. Then, the robot transports the hose to the hydrant as shown in Figure 6(f) and 6(g). Figure 6(h) shows the robot inserting the hose into the hydrant by moving its lower body as well as its right arm.

These experiments showed the flexibility and benefits of using a simulation tool to easily port planned manipulations to a physical robot.

VI. Discussion and Conclusion

Controlling a high DOF humanoid robot to autonomously perform a complex task is challenging. In this paper we have described a simulation environment for grasping and transport tasks that includes dynamics, and is easily ported to a real physical humanoid robot. The use of simulation is important as it allows stress testing on the methods chosen, and also allows us to build a library of physical movement tasks that can be re-used on other related tasks. This is important as it is intractable to simulate all possible tasks the humanoid may be required to perform, particularly in disaster scenarios. We hope to build a database of these movements which can then be indexed and parameterized by task specifications to extend the range of tasks of the humanoid.

Learning from the simulation and physical experiments, we are now working on optimizing the design of the HUBO robot. The current HUBO arm has six degrees of freedom. This poses limitations to trajectory planning. We are adding another DOF to the wrist so that we can benefit from the redundancy and have more flexibility in motion planning. Another focus is on the HUBO hand. The current robot hand is sensorless, which makes it difficult to have feedback control in a grasping or manipulation task. We are considering adding tactile and joint angle sensors to the hand so that we can use tactile feedback to track the motion of the hand and exploit some of the existing algorithms (e.g. [21], [22]) to increase the robustness of grasping and manipulation tasks.

While this paper addresses one component of the overall hose installment problem, we are continuing to integrate this phase of the task with the other phases described in section II-B. We are also interested in extending the methods described here to other physical humanoid platforms, which can be easily integrated into the simulation environment.

References

[1] IEEE, "Ieee." http://spectrum.ieee.org/automaton/robotics/humanoids/humanoid-robots-rise, 2012.

[2] T. Asfour, K. Yokoi, C. G. Lee, and J. Kuffner, "Humanoid robotics," *IEEE Robotics and Automation Magazine*, vol. 19, pp. 108–118, March 2012.

[3] N. Hudson, T. Howard, J. Ma, A. Jain, M. Bajracharya, S. Myint, C. Kuo, L. Matthies, P. Backes, P. Hebert, T. Fuchs, and J. Burdick, "End-to-end dexterous manipulation with deliberate interactive estimation," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 2371 –2378, may 2012.

[4] S. Srinivasa, D. Berenson, M. Cakmak, A. Collet Romea, M. Dogar, A. Dragan, R. A. Knepper, T. D. Niemueller, K. Strabala, J. M. Vandeweghe, and J. Ziegler, "Herb 2.0: Lessons learned from developing a mobile manipulator for the home," *Proceedings of the IEEE*, vol. 100, pp. 1–19, July 2012.

[5] J. A. D. Bagnell, F. Cavalcanti, L. Cui, T. Galluzzo, M. Hebert, M. Kazemi, M. Klingensmith, J. Libby, T. Y. Liu, N. Pollard, M. Pivtoraiko, J.-S. Valois, and R. Zhu, "An integrated system for autonomous robotics manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2955–2962, October 2012.

[6] M. Prats, P. Sanz, and A. del Pobil, "Task-oriented grasping using hand preshapes and task frames," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1794 –1799, april 2007.

[7] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng, "Robotic grasping of novel objects," in *Advances in Neural Information Processing Systems 19*, pp. 1209–1216, 2007.

[8] M. T. Ciocarlie and P. K. Allen, "Hand Posture Subspaces for Dexterous Robotic Grasping," *The International Journal of Robotics Research*, vol. 28, no. 7, pp. 851–867, 2009.

[9] M. Prats, P. Sanz, and A. del Pobil, "A framework for compliant physical interaction," *Autonomous Robots*, vol. 28, pp. 89–111, 2010. 10.1007/s10514-009-9145-8.

[10] J. Sturm, A. Jain, C. Stachniss, C. Kemp, and W. Burgard, "Operating articulated objects based on experience," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 2739 –2744, oct. 2010.

[11] M. Inaba, K. Okada, T. Yoshikai, R. Hanai, K. Yamazaki, Y. Nakanishi, H. Yaguchi, N. Hatao, J. Fujimoto, M. Kojima, S. Tokutsu, K. Yamamoto, Y. Kakiuchi, T. Maki, S. Nozawa, R. Ueda, and I. Mizuuchi, "Enhanced mother environment with humanoid specialization in irt robot systems," in *Robotics Research* (C. Pradalier, R. Siegwart, and G. Hirzinger, eds.), vol. 70 of *Springer Tracts in Advanced Robotics*, pp. 379–396, Springer Berlin Heidelberg, 2011.

[12] DARPA, "Drc." http://www.darpa.mil/Our_Work/TTO/Programs/DARPA_Robotics_Challenge.aspx, 2012.

[13] Wikipedia, "Hubo robot." http://en.wikipedia.org/wiki/HUBO, 2012.

[14] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *Robotics & Automation*, vol. 11, no. 4, pp. 110–122, 2004.

[15] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," Tech. Rep. CMU-RI-TR-08-34, Robotics Institute, Pittsburgh, PA, July 2008.

[16] J.-H. Kim and J. ho Oh, "Walking control of the humanoid platform khr-1 based on torque feedback control," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 1, pp. 623 – 628 Vol.1, april-1 may 2004.

[17] A. Miller and P. Allen, "Examples of 3d grasp quality computations," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 1240 –1246 vol.2, 1999.

[18] R. Suárez, M. Roa, and J. Cornella, "Grasp quality measures," tech. rep., Technical University of Catalonia, 2006.

[19] D. Berenson, S. S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, 2011.

[20] D. Berenson, S. Srinivasa, D. Ferguson, A. Collet Romea, and J. Kuffner, "Manipulation planning with workspace goal regions," in *IEEE International Conference on Robotics and Automation (ICRA '09)*, May 2009.

[21] K. Hsiao, L. Kaelbling, and T. Lozano-Prez, "Robust grasping under object pose uncertainty," *Autonomous Robots*, vol. 31, pp. 253–268, 2011. 10.1007/s10514-011-9243-2.

[22] H. Dang and P. K. Allen, "Tactile experience-based robotic grasping," in *Workshop on Advances in Tactile Sensing and Touch based Human-Robot Interaction, HRI*, March 2012.