Humanoid Task and Motion Planning using Backward-Forward Search

Caelan R. Garrett
MIT CSAIL
Cambridge, MA 02139 USA
caelan@csail.mit.edu

Michael X. Grey,
C. Karen Liu, and Aaron D. Ames
Georgia Institute of Technology
Atlanta, Georgia 30332 USA
mxgrey@gatech.edu,
karenliu@cc.gatech.edu, ames@gatech.edu

Andrea L. Thomaz
University of Texas at Austin
Austin, TX 78701 USA
athomaz@ece.utexas.edu

I. Introduction

The potential versatility of humanoid robots makes them promising candidates for a broad variety of applications, ranging from domestic assistance and commercial hospitality to industrial labor and disaster relief. While modern humanoids are still often teleoperated, we ultimately wish for them to be able to autonomously perform these everyday tasks with minimal instruction in order for them to be maximally economical. To do this, humanoids must be able to independently solve task and motion planning problems wherein they determine not only the sequence of high-level actions that will achieve their task but also the joint motions that are necessary to perform it in the real world. However, solving task and motion planning problems is particularly challenging for humanoids due to their complexity. Additionally, the biped nature of humanoids requires footstep planning which can be computationally expensive.

We propose a system for solving humanoid manipulation problems that balances the trade-off between efficiency and completeness. Our system uses the hybrid backward-forward (HBF) planning algorithm as a task-planner along with humanoid manipulation primitives for pick, place, move, and press actions. These primitives are particularly designed to be efficient enough to sample many times during the task-planning search while still powerful enough to produce a wide range of robot behaviors. We compromise by searching through continuous subspaces of the configuration, subspaces which are guaranteed to be reachable from the robot's constraint manifold, allowing the robot to reason abstractly about its task while guaranteeing the feasibility of its plan.

We empirically evaluate this system on three complex pick-and-place problems. In particular, the latter two problems involve gate obstacles that the robot must toggle in order to walk throughout the environment. Our experiments show that our humanoid samplers are able to quickly operate while still faithfully representing the capabilities of the robot allowing HBF to efficiently solve these problems.

II. HBF Algorithm

HBF is an algorithm for solving high-dimensional planning problems in hybrid state spaces [1]. It uses an approximate

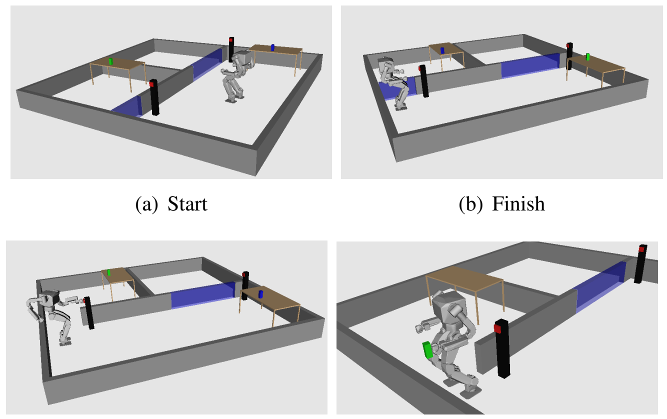
(a) Start (b) Finish
(c) Lowering force field (d) Sidestepping through passage

Fig. 1. Second Scenario: Robot must swap the table that each object is sitting on while dealing with a force field obstruction.

backwards search to focus the sampling of successor actions in forward, state space heuristic search. When previously applied to mobile manipulation problems using a PR2 robot, it was able to solve long-horizon planning problems in around one minute. In order to apply HBF to our humanoid manipulation problems, we must represent the abstract manipulation actions the robot may perform (such as Pick and Move) and specify samplers that produce instantiations of these actions as fully parameterized manipulation primitives.

Garrett et al. represent abstract actions such as Pick, Place, Move, and MoveHolding as *action templates*, parameterized sets of action constraints (con) and effects (eff) [1]. We introduce the PressDown and PressRelease actions for pressing buttons. In our experiments, we explore problems involving "force fields" as obstacles which constrain the movement of the robot. While the force fields cannot be manipulated directly themselves, each force field is connected to button k which can toggle the state of certain force fields m between a disabled value $f_m =$ **None** and an enabled value $f_m = p_m^0$ where p_m^0 is the fixed initial pose of force field m. The buttons themselves have an on state $b_k =$ **Active** and an off state $b_k =$ **Inactive**. PressDown activates a button and PressRelease deactivates a button. Depending on the “wiring” of buttons and force fields given in a problem, activating a button may enable, disable, or not affect a force field.

This description of a PressRelease action template resembles the Pick template. A single trajectory τ is sufficient to represent the move from standing robot configuration q to press the button and return to q. Additionally, k is the label for the button to be pressed. Releasing button k will disable some force fields and enable other force fields. Note that PressDown is defined similarly.

PressRelease(q, k, τ):

$$\begin{aligned}
\text{con}: \quad & r, h, b_k = q, \textbf{None}, \textbf{Active} \\
& o_i \in \text{c-free-poses}_i(\tau)\ \forall i \\
\text{eff}: \quad & b_k = \textbf{Inactive} \\
& f_m = \textbf{None}\ \forall m \in \text{disable-force-fields}(k) \\
& f_n = p_n^0\ \forall n \in \text{enable-force-fields}(k)
\end{aligned}$$

III. Humanoid Manipulation Primitives

Now that we have identified HBF’s task-level action templates, we must produce samplers that can efficiently produce instantiations of these Move, Pick, Place, MoveHolding, PressRelease, and PressDown actions. This requires developing whole body inverse kinematics solvers, standing motion planners, and whole body motion planners for humanoid robots. However, this is more challenging to do for humanoid robots than standard mobile manipulators for several reasons. First, humanoid robots tend to be much higher dimensional. A one-armed robot manipulator on a mobile platform tends to have 10-11 degrees of freedom. Conversely, a humanoid platform tends to have anywhere from 30-50 degrees of freedom. This larger dimensionality makes it much more time consuming to perform standard robotics procedures such as solving inverse kinematics. Second, whereas mobile manipulators are only restricted by joint limits and collision constraints, humanoid robots also have to handle balance constraints and end effector constraints. Finally, many standard mobile manipulators use a holonomic, wheeled base while humanoid robots are bipedal, requiring footstep planning. For the purpose of this extended abstract, we just describe the walking motion planning.

In order to sample Move and MoveHolding actions in our representation, we need a way of producing walking trajectories. Rigorously planning out footsteps tends to require a significant amount of computational time [2]. In our implementation, it takes on the order of thirty seconds to one minute per call. HBF must produce a substantial number movement actions during its search as it moves to manipulate objects from different states. Many of these movement actions do not ultimately end up in the final plan. Thus, it would be incredibly cumbersome to generate full footstep trajectories during the planning process. At the same time, totally ignoring the walking motion planning by assuming a trajectory exists can result in an substantially incomplete planner on problems which impose constraints on the reachable walking configurations.

To address this, we simplify the walking problem by turning it into a route finding problem. This allows us to ignore the balancing and end effector constraints that are required while generating a standing trajectory. All we are left with is a standard collision-free path finding problem, so we can employ traditional RRT-connect[3] in a 3D domain (see Figure 2) for blazingly fast planning times.

However, there is a catch: The full motion that the robot must go through during a walking cycle is not captured with a standard three-dimensional RRT. There are also torso sway and leg motion factors that must be taken into account. We do this by constructing a collision geometry that approximates the swept volume that the robot might move through during an arbitrary walk cycle. Performing collision checks against this extended geometry guarantees that the robot will be able to find a feasible footstep plan along any path that is swept by it. Incorporating two translational dimensions and the yaw dimension in the plan allows the robot to make use of its sidestepping capabilities, which can be useful for squeezing through tight spaces, as seen in Figure 1(d). Since we know that the generated routes will allow for feasible footstep plans, we can defer computing the footsteps until HBF produces a final plan. This saves us from performing expensive footstep computations on unused subplans.

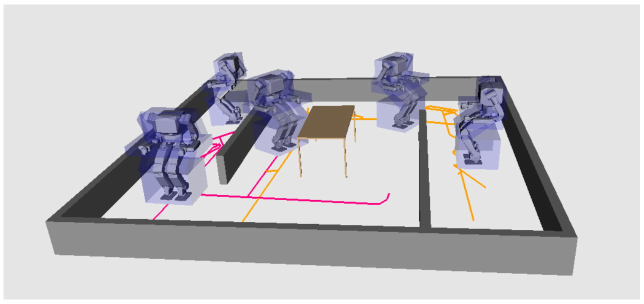
Fig. 2. The route planning uses a 3D RRT: two dimensions for translation and one dimension for yaw. Overlapping edges in the visualization are an artifact of projecting away the yaw dimension. We treat the robot as holonomic because it is capable of taking a step in any direction at any time. This allows the RRT search to be unconstrained and therefore extremely fast.

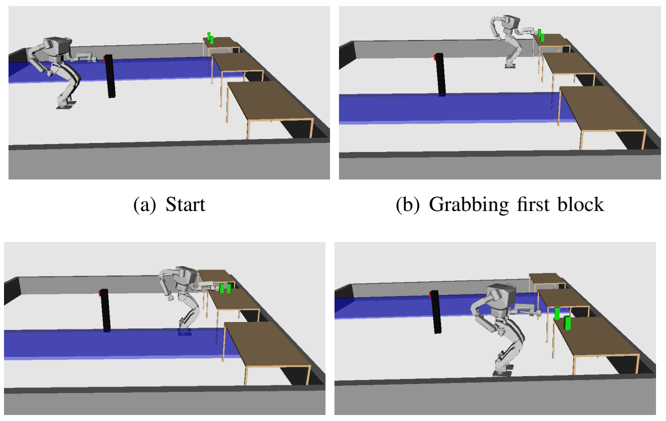
(a) Start (b) Grabbing first block (c) Moved both to center table (d) Finish

Fig. 4. Third Scenario: Robot must move both blocks across the room while switching the force field back and forth.

P	Deferred Route Motion Planning				Standard RRTs				Multi-Query Star Roadmap			
	%	time	len	foot	%	time	len	foot	%	time	len	foot
1	55	251 (121)	24 (0)	433	80	199 (26)	24 (0)	549	80	187 (34)	24 (0)	701
2	0	600 (0)	- (-)	-	95	266 (19)	27 (0)	619	100	144 (13)	27 (0)	1611
3	0	600 (0)	- (-)	-	100	215 (17)	13 (0)	562	100	85 (11)	13 (0)	636

Fig. 3. Manipulation experiment results over 20 trials.

For an additional performance boost, at the expense of completeness on some problems, we experimented with a meta motion planning roadmap called a star roadmap. When tasked with finding a trajectory from q to q', if not already cached, it computes RRT trajectories from a fixed center configuration q_0 (such as the initial robot configuration) to both q and q'. The resulting trajectories are cached as edges in the star graph. The returned solution is the reversed trajectory from q to q_0 concatenated with the trajectory from q_0 to q'.

IV. Experiments

We tested the effectiveness of our system on three diverse scenarios. These scenarios focus on pick-and-place tasks but also include obstacles that require the robot to alter its environment in order to achieve its goals. Obstacles come in two forms: small manipulatable items that obstruct the robot's ability to reach for goal items, and "force field" barriers that are controlled by switches in the environment. The "force fields" can be thought of as automated doors that can be operated by pressing a button.

a) First Scenario: There are two tables in the environment. One table is empty while the other is covered in blocks. The robot's objective is to move the two blue blocks to the other table. It is free to move the red blocks though.

b) Second Scenario: There are two tables and two force fields. One force field is blocking the way to one of the tables while the other force field leads nowhere. The robot's objective is to move each block to the table that it is not currently sitting on. Also, courtesy dictates that the robot must return itself and all force fields back to their original states. This is an example of a non-monotonic problem which requires the robot to undo some of its goals in order to accomplish others. See Figure 1.

c) Third Scenario: There are three tables. A button in the center of the room is able to swap a force field from one side of the room to the other side. The robot must move both of the green blocks from the far side of the room to the near side of the room while switching the force field as needed. To do this, the robot must temporarily place each block on the middle table in order to change the force field which results in puzzle-like behavior. See Figure 4.

We tested three versions of the resulting system that each handle route planning differently.

d) Deferred Route Motion Planning: This version defers all walking motion planning, not just footstep computation, until HBF finds a solution by assuming each trajectory is feasible. If when HBF finds a solution the deferred motion planning problems are not feasible, the algorithm fails.

e) Standard RRT: This version uses the standard strategy of calling a new RRT to sample each movement action.

f) Multi-Query Star Roadmap: This version uses the multi-query star roadmap to answer motion planning queries.

Each experiment had a 10 minute timeout. There were 20 trials per problem all conducted on a single 1.87GHz Intel Core i7 processor. Each entry in figure 3 reports the success percentage (%) as well as the median and median absolute deviation (MAD) of the runtime, resulting symbolic plan length, and the post-processing time it took to generate footsteps for a single run. We use median-based statistics to be robust against outliers. The statistics for trials that failed to find a solution are included in the entries. Thus, entries with a runtime of 600 and MAD of 0 did not solve any trial.

The multi-query star roadmap proved more efficient than the normal RRT strategy as it was able to reduce the number of new RRT calls. Note that the deferred route motion planning strategy failed to solve problems two and three at all because they involve force fields which need to be deactivated. Thus, the route planning compromise of approximating the walking robot using a swept volume is able to provide sufficient information for HBF to solve these problems while being efficiently computable and later resulting in a valid footstep trajectory. Finally, although the post-processing footstep computation time is still large, the footsteps themselves can be computed online while the humanoid executes the plan because our method guarantees that a satisfying solution exists.

References

[1] Caelan Reed Garrett, Tomas Lozano-Perez, and Leslie Pack Kaelbling. Backward-forward search for manipulation planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015. URL http://lis.csail.mit.edu/pubs/garrett-iros15.pdf.

[2] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue. Online footstep planning for humanoid robots. In *International Conference on Robotics and Automation (ICRA)*, volume 1, pages 932–937 vol.1, Sept 2003. doi: 10.1109/ROBOT.2003.1241712.

[3] J. J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *International Conference on Robotics and Automation (ICRA)*, volume 2, pages 995–1001 vol.2, 2000. doi: 10.1109/ROBOT.2000.844730.