Humanoid Manipulation Planning using Backward-Forward Search*

Michael X. Grey[1], Caelan R. Garrett[2],
C. Karen Liu[1], Aaron D. Ames[1], and Andrea L. Thomaz[3]

***Abstract*— This paper explores combining task and manipulation planning for humanoid robots. Existing methods tend to either take prohibitively long to compute for humanoids or artificially limit the physical capabilities of the humanoid platform by restricting the robot's actions to predetermined trajectories. We present a hybrid planning system which is able to scale well for complex tasks without relying on predetermined robot actions. Our system utilizes the *hybrid backward-forward* planning algorithm for high-level task planning combined with humanoid primitives for standing and walking motion planning. These primitives are designed to be efficiently computable during planning, despite the large amount of complexity present in humanoid robots, while still informing the task planner of the geometric constraints present in the problem. Our experiments apply our method to simulated pick-and-place problems with additional gate constraints impacting navigation using the DRC-HUBO1 robot. Our system is able to solve puzzle-like problems on a humanoid within a matter of minutes.**

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. Humanoid robots are high-dimensional systems with low-dimensional constraint manifolds. This makes their configuration spaces difficult to explore comprehensively and thus their capabilities hard to fully utilize. Methods that exhaustively search through their configuration spaces when planning a manipulation are prohibitively slow. To circumvent this problem, many approaches preprogram a set of manipulation trajectories that can be used, thus drastically simplifying a humanoid's domain. However, this is problematic in itself because it often discards a large amount of manipulation capabilities that are needed to robustly solve everyday problems. Additionally, the biped nature of humanoids requires footstep planning which can be computationally expensive.

In this paper, we propose a system for solving humanoid manipulation problems that balances this 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 do not discretize the physical actions that are available to the robot, but we also do not search strictly within the valid configuration space of the robot. Instead, we compromise by searching through continuous subspaces of the configuration—subspaces which are guaranteed to be reachable from the robot's constraint manifold. This allows the robot to reason abstractly about its task while guaranteeing that its plan of action will be feasible.

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.

A. Related Work

Due to the complexity of humanoid robots, their technical challenges are usually decomposed into various independent fields of study. For example, Navigation Among Movable Obstacles (NAMO) [1] and footstep planning [2] are often studied as separate problems, even though navigating around and moving objects in an environment requires the robot to perform footsteps. NAMO solvers typically avoid footstep planning by projecting the problem onto a 2D grid and providing the robot with a discrete set of predetermined footstep actions. While this approach may be fast and capable of running online, it discards all of the more complex behaviors that a humanoid may be able to utilize for accomplishing its goals. For example: a humanoid may be able to crawl under a heavy obstacle rather than move the obstacle out of the way, but this would elude a NAMO planner which projects the environment onto a planar grid.

*This work was supported in part by DARPA grant D15AP00006, NSF grants 1420927 and 1523767, ONR grant N00014-14-1-0486, and ARO grant W911NF1410433

[1]Georgia Institute of Technology, Atlanta, GA 30332 USA mxgrey@gatech.edu, karenliu@cc.gatech.edu, ames@gatech.edu

[2]Computer Science and Artificial Intelligence Laboratory at Massachusetts Institute of Technology, Cambridge, MA 02139 USA caelan@csail.mit.edu

[3]University of Texas at Austin, Austin, TX 78701 USA athomaz@ece.utexas.edu

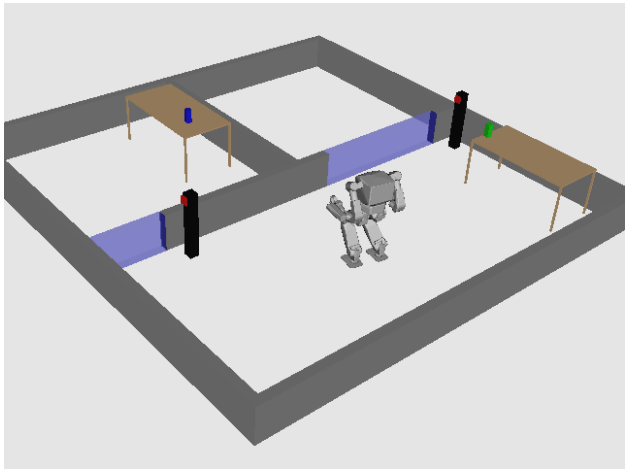


Fig. 1. Example scenario: The robot must swap the blocks that are on the tables. A "force field" gate blocks the robot's path, and the robot must press a switch to disable it.

On the other end of the spectrum are methods which search through the full configuration space of the problem [3]. These methods are complete and able to fully exploit the versatility of the humanoid platform in theory. However, they suffer from extremely long run times (minutes to hours) in static environments. This is not likely to scale well if a task requires the robot to alter its environment, such as moving an obstacle out of the way of its goal.

Prior work on integrated task and motion planning mostly focuses on standard mobile manipulators applied to pick-and-place tasks [4], [5], [6]. These methods are able to efficiently project the geometry and kinematics of pick-and-place problems into a symbolic search which ultimately produces a sequence of valid motion plans.

II. HBF Algorithm

HBF is an algorithm for solving high-dimensional planning problems in hybrid state spaces [6]. It uses an approximate backward 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.

III. Action Representation

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) [6]. Our representation of actions will be similar apart from two differences. First, rather than strictly separate the motion from the contact, we couple each Pick and Place action with the standing trajectories used to pick or place the object. Consequently, each Move and MoveHolding can then just be used to express walking trajectories. Consider the following Pick action as defined using the action template structure. Its parameters are a standing robot configuration q, grasp robot configuration q', manipulated object label j, grasp transform γ, approach trajectory τ_1, and departure trajectory τ_2. There are implicit, permanent constraints on the parameters such that τ_1 is a valid trajectory from q to q' as well as τ_2 is a valid trajectory from q' to q while holding object o_j with grasp transform γ.

Pick$(q, q', j, \gamma, \tau_1, \tau_2)$:

con : $r, h, o_j = q, \textbf{None}, \text{Pose}(q', \gamma)$
$o_i \in \text{collision-free-poses}_i(\tau_1) \cap$
$\text{collision-free-poses-holding}_i(\tau_2, j, \gamma)\ \forall i \neq j$

eff : $h, g, o_j = j, \gamma, \textbf{None}$

Notice that in its effects, instead of updating the value of o_j to its true pose, Pick assigns it **None** which indicates that the o_j is not placed. The true pose of the object can still be derived from Pose(q, γ) using forward kinematics when needed. As a consequence, our MoveHolding actions now need not update o_j because the pose of object j is always implicitly available. Our Place actions are defined using similar changes.

The motivation for this representational difference is to be able to easily reconsider pick and place actions in the planning process. Although the planner may not immediately pursue an action sampled in the search, it may prove useful later on. Because sampling the approach and departure trajectories is particularly time-consuming for a humanoid robot, it is advantageous to try to reuse as many previously sampled actions as possible rather than always resample.

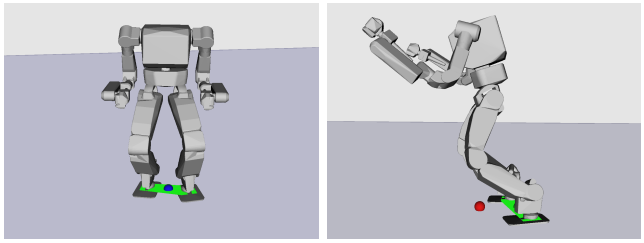
Additionally, we introduce the PressDown and PressRelease actions for pressing buttons. In our experiments, we explore problems involving "force fields" as obstacles which constrain the movement of the robot (see Figure 1). While the force fields cannot be manipulated directly themselves, each force field is connected to button k which can toggle the state of certain force fields m between a disabled value $f_m = \textbf{None}$ and an enabled value $f_m = p^0_m$ where p^0_m is the fixed initial pose of force field m. The buttons themselves also have an on state $b_k = \textbf{Active}$ and an off state $b_k = \textbf{Inactive}$ where PressDown activates the button and PressRelease deactivates the buttons. Depending on the "wiring" of buttons and force fields, 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):

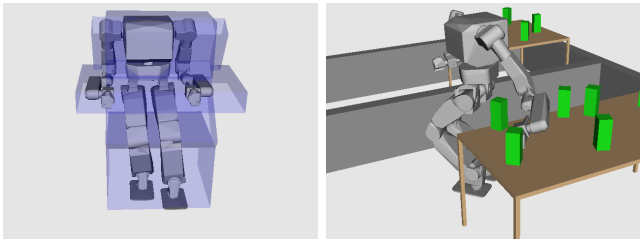
con : $r, h, b_k = q, \textbf{None}, \textbf{Active}$
$o_i \in \text{collision-free-poses}_i(\tau)\ \forall i$

eff : $b_k = \textbf{Inactive}$
$f_m = \textbf{None}\ \forall m \in \text{disable-force-fields}(k)$
$f_n = p^0_n\ \forall n \in \text{enable-force-fields}(k)$



(a) Balanced Configuration (b) Unbalanced Configuration

Fig. 2. The green polygon is the "support polygon". The blue dot in 2(a) represents the balanced ZMP while the red dot in 2(b) represents the unbalanced ZMP.



(a) Start of a left-foot forward step (b) Dexterous manipulation task

Fig. 3. Visualizations of collision geometries (a) during walking and (b) during manipulation. A "safe-zone" is given to the robot while walking to ensure that it will be able to find a collision-free footstep plan along its route. But this "safe-zone" is not used during manipulation, so the robot is free to be dexterous.

IV. Humanoid Manipulation Primitives

Now that we have identified the task-level action templates that HBF will use, we need to 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.

We begin by describing how we incorporate the balance and end-effector constraints in our manipulation primitives.

1) Balance Constraints: A common heuristic for ensuring that a humanoid is balanced, is for its Zero Moment Point (ZMP) [7] to remain underneath its "support polygon" [8]. For simplification purposes, we assume quasi-static motion while generating a plan. In other words, we assume the robot is moving slowly enough that its ZMP lines up with the downward projection of its center of mass. This allows us to leave velocity out of the configuration space while planning. An illustration of a balanced versus unbalanced configuration can be seen in Figure 2. Our algorithm for finding balanced configurations can be found in lines 20-24 of Algorithm 1.

2) End-Effector Constraints: A standard mobile manipulator performing a one-handed grasping task only needs to consider a 6-dimensional pose constraint for one end effector. When a humanoid is generating a standing trajectory, it additionally needs to consider a 6-dimensional pose constraint for each foot. We express these end effector constraints as Task Space Regions (TSR) [9]. For an end effector that is fully constrained, we set all the boundary parameters to zero (or $\pm\epsilon$, some small tolerance to help with convergence).

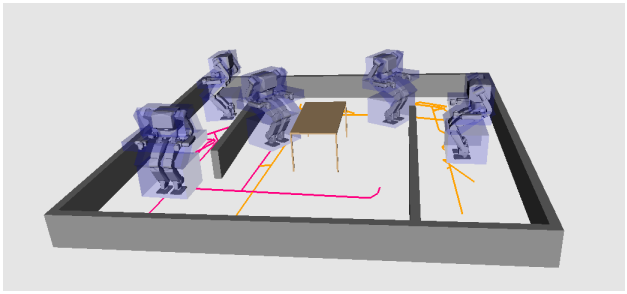


Fig. 4. The route planning uses a 3D RRT: two dimensions for translation and one dimension for yaw. Overlapping edges in the visualization are an artifact of projecting away the yaw dimension. We treat the robot as holonomic because it is capable of taking a step in any direction at any time. This allows the RRT search to be unconstrained and therefore extremely fast.

A. Route 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. In our implementation, it takes on the order of thirty seconds to one minute per call. HBF must produce a substantial number of 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 a 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[10] in a 3D domain (see Figure 4) 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 (see Figure 3). Performing collision checks against this extended geometry guarantees that the robot will be able to find a feasible footstep plan along any path that it sweeps. 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.

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 subplans that will never get used.

Multi-Query Star Roadmap: 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*. The roadmap is a star graph where the center is an arbitrarily chosen configuration q_0 (such as the initial robot configuration). Each time a new motion planning query q, q' is made, if q or q' is not already contained in the star roadmap, a standard motion planner (i.e. our RRT) computes trajectories from q_0 to q or from q_0 to q' respectively. These trajectories become new edges $(q_0 \rightarrow q)$ and $(q_0 \rightarrow q')$ in the star graph. The returned trajectory is then $q_0 \rightarrow q$ reversed and concatenated with $q_0 \rightarrow q'$. Note that this assumes the trajectories are reversible. This roadmap is simple to implement and can be helpful for task and motion planning problems where we may have already indirectly found a plan from q to q' earlier in the search. In particular, the number of trajectories computed is always linear in the number of reached configurations instead of quadratic. However, it is incomplete because in some cases, it may be necessary to obtain a path directly from q to q' without moving through q_0. Still, it performs reliably on problems which effectively have only one homotopic class.

B. Grasping Configurations

To produce Pick, Place, PressDown, and PressRelease actions, we require the ability to sample grasping configurations. Note that although PressDown and PressRelease never actually grasp the button, we treat the button press configuration as a grasp configuration.

For a grasping configuration to be valid, it must satisfy all the same constraints mentioned in section IV-.1 through IV-.2, except that the feet are less constrained: x-/y-translation and yaw have no limits (i.e. their TSR boundary parameters are $\pm\infty$).

There are two additional constraints that a grasping configuration must satisfy in order to be considered "valid". First, the route planner must be able to approach it. What we mean by "approach" is that it must be possible for the walking collision geometry (see Figure 3) to stand at the grasp configuration without being in collision with the environment. Otherwise, there is no guarantee that it will be possible for the robot to walk into the foot placements needed for the grasping configuration.

Secondly, a grasping configuration is only considered valid if the robot can move itself into a "walk ready" configuration without moving its feet. If a standing trajectory cannot be found (within a reasonable time limit) from the grasping configuration to a stance that permits walking, then we must assume that the grasping configuration cannot be reached.

Finding valid grasp configurations is perhaps the most computationally expensive aspect of our entire planning process. To make it as computationally efficient as possible, we employ some important improvements to standard whole body IK methods.

1) Computing Whole Body Inverse Kinematics: We primarily use built-in features of the DART[1] [11] software package. DART solves whole body inverse kinematics by hierarchically combining several constrained optimization problems into one. Each constraint (grasping, stance feet, balance) offers its own error gradient which get superimposed on each other to produce the net error gradient. When constraints are put into a hierarchy, the lower priority constraint error gradients get projected through the null space of the higher priority constraint error Jacobians.

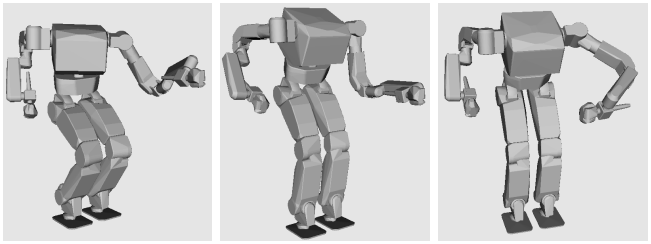
Analytical inverse kinematics solutions are available for each limb of the DRC-HUBO1 robot. DART's whole body inverse kinematics solver can leverage the analytical solutions for each limb to considerably speed up the whole body IK. Some pseudo-code that shows how this can be done is given in Algorithm 1.

Often whole body IK methods are applied in local optimization problems [12][13][14] or local control algorithms [15][16][17] because iterative whole body IK methods tend to be effective at correcting small violations of many simultaneous constraints. However, we will need to find valid grasping configurations across a broad domain as seen in Figure 6, which puts the whole body IK solver at risk of getting caught in local minima.

To avoid local minima traps, we start off the whole body IK problem by localizing it. We begin with a seed configuration, adjust the height, pitch, and roll of the end effector to match the target grasp pose using whole body IK without concern for x-/y-translation or yaw. Once that has converged, we *then* directly alter the root transform of the robot in x-/y-translation and yaw to place the end effector at its target. The whole body IK is then briefly applied once more to wash away small errors caused by numerical imprecision. The procedure is laid out in Algorithm 2. This localized solving approach exploits the fact that the robot's stance and balance constraints are invariant to the x-/y-translation and yaw of the overall robot. Therefore, we defer those dimensions of the IK problem until after we locally solve the dimensions that the stance and balance constraints do depend on: z-translation, roll, and pitch.

2) Configuration seeds: The constraint manifold for a balanced humanoid is very low dimensional compared to its

[1]More information on DART can be found at http://dartsim.github.io/



(a) (b) (c)

Fig. 5. Examples of useful seed configurations that were given to the whole body inverse kinematics for finding left-handed grasping configurations.

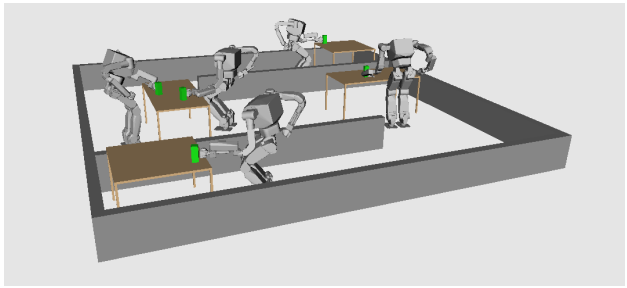


Fig. 6. The whole body inverse kinematics solver needs to be applicable over a wide region.

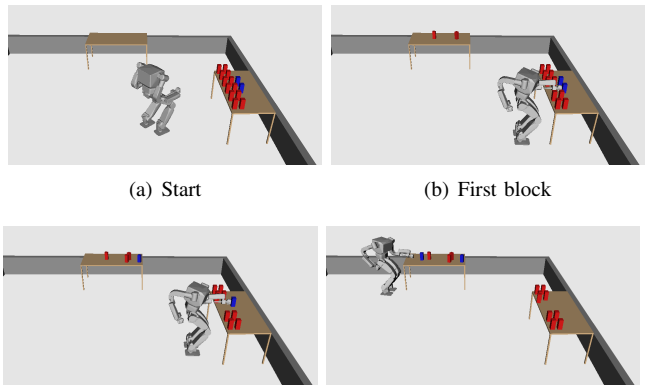
overall configuration space. This means that the probability of randomly sampling configurations that are anywhere near the constraint manifold is close to zero. This is especially problematic because whole body IK is vulnerable to getting caught in local minima, so the further a random configuration is from the constraint manifold, the less likely it is that the whole body IK will be able to converge onto the constraint manifold from it.

To avoid this issue, we can start from seed configurations. Seed configurations are predetermined configurations that are likely to be useful for the whole body IK problems that the robot encounters. Seeds can be computed offline by a procedure of aggressive sampling and optimization or be provided by a knowledgeable user. They can also be saved and loaded from databases that persist between problem encounters [18]. Some examples of seed configurations that were used for this paper can be seen in Figure 5.

V. 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.

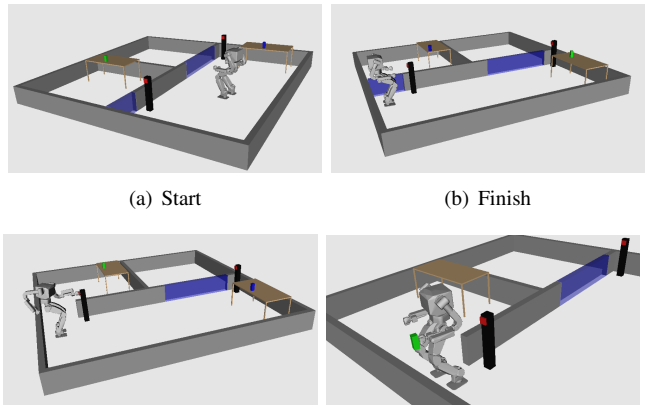
a) First Scenario: There are two tables in the environment. One table is empty while the other is covered in blocks. The robot's objective is to move the two blue blocks



(a) Start (b) First block

(c) Second block (d) Finish

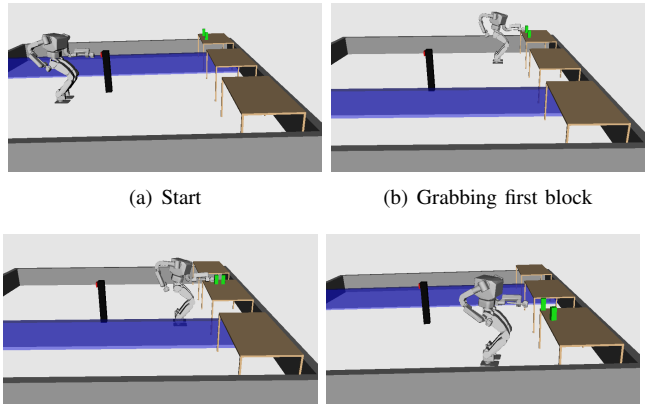
Fig. 7. First Scenario: Robot must move the blue blocks to the other table.



(a) Start (b) Finish

(c) Pressing switch to lower force field (d) Sidestepping through narrow passage

Fig. 8. Second Scenario: Robot must swap the table that each object is sitting on while dealing with a force field obstruction.



(a) Start (b) Grabbing first block

(c) Moved both to center table (d) Finish

Fig. 9. Third Scenario: Robot must move both blocks across the room while switching the force field back and forth.

Algorithm 1 Iterative-Analytical Hybrid Whole Body IK

function SOLVEWHOLEBODYIKFORGRASP(q, T_{grasp})
  $t \leftarrow 0$
  $\gamma \leftarrow$ GetWeights()
  $\lambda \leftarrow$ GetDampingCoefficient()
  $\epsilon \leftarrow$ GetTolerance()
  $maxT \leftarrow$ GetMaximumIterations()
  do
    $\Delta q \leftarrow \mathbf{0}$
    for f in StanceFeet() **do** ▷ Flat Foot Constraints
      $T_{foot} \leftarrow$ ComputeFootFK(f, $q + \Delta q$)
      $T_{goal} \leftarrow$ ProjectToGround(T_{foot})
      $q_f \leftarrow$ AnalyticalIKForFoot(f, T_{goal}, $q + \Delta q$)
      $T_{foot} \leftarrow$ ComputeFootFK(f, q_f)
      $T_{goal} \leftarrow$ ProjectToGround(T_{foot})
      $\Delta x_f \leftarrow$ ComputeScrew(T_{goal}, T_{foot})
      $J_f \leftarrow$ ComputeJacobianForFoot(f, q_f)
      $q_f \leftarrow q_f + \gamma J_f^T (\lambda I + J_f J_f^T)^{-1} \Delta x_f$
      $\Delta q \leftarrow q_f - q$
    end for
    ▷ Balancing Constraint
    $x_{com} \leftarrow$ ComputeCenterOfMass($q + \Delta q$)
    $x_p \leftarrow$ ComputeSupportPolygonCenter($q + \Delta q$)
    $\Delta x_{com} \leftarrow x_p - x_{com}$
    $J_{com} \leftarrow$ ComputeCOMJacobian($q + \Delta q$)
    $\Delta q \leftarrow \Delta q + J_{com}^T (\lambda I + J_{com} J_{com}^T)^{-1} \Delta x_{com}$
    ▷ Grasping Constraint
    $q_h \leftarrow$ AnalyticalIKForHand(T_{grasp}, $q + \Delta q$)
    $T_{hand} \leftarrow$ ComputeHandFK(q_h)
    $\Delta x_h \leftarrow$ ComputeScrew(T_{grasp}, T_{hand})
    $J_h \leftarrow$ ComputeJacobianForHand(q_h)
    $q_h \leftarrow q_h + \gamma J_h^T (\lambda I + J_h J_h^T)^{-1} \Delta x_h$
    $\Delta q \leftarrow q_h - q$
    ▷ Conclude this iteration
    $q \leftarrow q + \Delta q$
    $t \leftarrow t + 1$
  while norm(Δq) $> \epsilon$ and $t < maxT$
  if norm(Δq) $> \epsilon$) **then**
    return None
  end if
  return q
end function

Algorithm 2 Localized Whole Body IK

function FINDGRASPINGCONFIGURATION(T_{grasp})
  $q \leftarrow$ GetSeedConfiguration()
  $q \leftarrow$ RandomizedPerturbation(q)
  $T_{hand} \leftarrow$ ComputeHandFK(q)
  $z_{desired} \leftarrow$ GetTranslationZ(T_{grasp})
  SetTranslationZ(T_{hand}, $z_{desired}$)
  $R_{xy} \leftarrow$ GetRotationXY(T_{grasp})
  SetRotationXY(T_{hand}, R_{xy})
  $q \leftarrow$ SolveWholeBodyIKForGrasp(q, T_{hand})
  if q is **None then** ▷ Failure to converge
    return None
  end if
  $T_{hand} \leftarrow$ ComputeHandFK(q)
  $T_{root} \leftarrow$ GetRootTransform(q)
  $T_{root} \leftarrow T_{grasp} T_{hand}^{-1} T_{root}$
  $q \leftarrow$ SetRootTransform(q, T_{root})
  return SolveWholeBodyIKForGrasp(q, T_{grasp})
end function

to the other table. It is free to move the red blocks around however it chooses. See Figure 7.

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 8.

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 9.

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 the deferred motion planning problems are not feasible when HBF finds a solution, then 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. We use deferred greedy best first search [19] as the HBF search control in our experiments which gives a slight increase in performance over enforced hill climbing search.

Each entry in figure 10 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 accompanying video simulates the DRC-HUBO1 executing a solution for each problem.

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. The footstep computation time tends

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. 10. Manipulation experiment results over 20 trials. Multiple trials are used because the results of randomized planners are not deterministic.

to be larger for multi-query star roadmap simply because its chaining of trajectories produces longer trajectories. An additional post-processing method could further smooth the chained trajectories before performing footstep planning. 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.

VI. Conclusions

This work has presented a new system for combining task and manipulation planning on humanoid robots. The method presented is faster than existing methods by orders of magnitude, although probabilistic completeness for edge cases is sacrificed in favor of speed. It is capable of solving tasks that constrain the reachable walking configurations of the robot while being more versatile than high-level planning methods that utilize predetermined primitive trajectories.

While the generated plans may be physically feasible for the robot, they are not necessarily of good "quality". The randomization factor of the motion planner tends to produce trajectories that are inefficient or close to unstable regions. In future work, to improve the quality of the plans that are generated, we will use the feasible plans to inform trajectory optimization routines and online control methods [20] [21] [22], ensuring that the actions performed by the robot are stable and efficient in addition to feasible.

References

[1] M. Stilman, K. Nishiwaki, S. Kagami, and J. J. Kuffner, "Planning and executing navigation among movable obstacles," *Advanced Robotics*, vol. 21, no. 14, pp. 1617–1634, 2007.

[2] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Online footstep planning for humanoid robots," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 1, Sept 2003, pp. 932–937 vol.1.

[3] K. Hauser and J.-C. Latombe, "Integrating task and prm motion planning: Dealing with many infeasible motion planning queries," in *ICAPS Workshop on Bridging the gap between task and motion planning*, 2009, pp. 19–23.

[4] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *IEEE Conference on Robotics and Automation (ICRA)*, 2014.

[5] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: An efficient heuristic for task and motion planning," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

[6] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "Backward-forward search for manipulation planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015. [Online]. Available: http://lis.csail.mit.edu/pubs/garrett-iros15.pdf

[7] M. VUKOBRATOVIĆ and B. BOROVAC, "Zero-moment point — thirty five years of its life," *International Journal of Humanoid Robotics*, vol. 01, no. 01, pp. 157–173, 2004.

[8] E. Yoshida, O. Kanoun, C. Esteves, and J.-P. Laumond, "Task-driven support polygon reshaping for humanoids," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. IEEE, 2006, pp. 208–213.

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

[10] 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 vol.2.

[11] C. K. Liu and S. Jain, "A short tutorial on multibody dynamics," Georgia Institute of Technology, School of Interactive Computing, Tech. Rep. GIT-GVU-15-01-1, 08 2012.

[12] M. Gienger, H. Janssen, and C. Goerick, "Task-oriented whole body motion for humanoid robots," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, Dec 2005, pp. 238–244.

[13] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," in *ACM transactions on graphics (TOG)*, vol. 23, no. 3. ACM, 2004, pp. 522–531.

[14] K. Yamane and Y. Nakamura, "Natural motion animation through constraining and deconstraining at will," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 3, pp. 352–360, July 2003.

[15] T. Sugihara and Y. Nakamura, "Whole-body cooperative balancing of humanoid robot using cog jacobian," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3, 2002, pp. 2575–2580 vol.3.

[16] 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.

[17] M. Mistry, J. Nakanishi, G. Cheng, and S. Schaal, "Inverse kinematics with floating base and constraints for full body humanoid robot control," in *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, Dec 2008, pp. 22–27.

[18] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3671–3678.

[19] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[20] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," in *accepted by IEEE Conference on Robotics and Automation*, 2016.

[21] C. M. Hubicki, A. Hereid, M. X. Grey, A. L. Thomaz, and A. D. Ames, "Work those arms: Toward dynamic and stable humanoid walking that optimizes full-body motion," in *accepted by IEEE Conference on Robotics and Automation*, 2016.

[22] A. D. Ames and M. Powell, "Towards the unification of locomotion and manipulation through control lyapunov functions and quadratic programs," in *Control of Cyber-Physical Systems*. Springer, 2013, pp. 219–240.