A Policy Representation for Humanoid Manipulation Tasks based on Kinematic Constraints*

Michael Grey[1]

I. Introduction

In the interest of quickly generating viable robot motions that require minimal real time control and sensing for success, we build on the task constrained motion planning [1] and Constrained Bi-directional Rapidly exploring Random Tree (CBiRRT) [2] framework. CBiRRT is a probabilistically complete motion planning algorithm which is able to rapidly generate motion plans in high dimensional space while respecting task constraints. CBiRRT works by growing random trees from a set of start nodes to a set of goal nodes. Each node must satisfy the task's feasibility constraints and cannot be further than some (user-defined) distance from its parent node. Trees are grown by alternating between random extensions (where a branch is grown directly toward a randomly selected point in configuration space) and "connection" attempts which try to immediately close the gap between a start and goal tree. In practice, this approach offers an effective balance between exploration and exploitation. Since humanoid robots are high degree of freedom systems with a wide variety of constraints, CBiRRT is well suited for planning in this domain.

However, one pitfall of CBiRRT is that its speed drops off significantly when the constraint manifold is a much lower dimension than the configuration space. To address this, we use CBiRRT while assuming the robot undergoes quasistatic motion. This allows us to plan a path through joint space without needing to consider joint velocities. Determining appropriate joint velocities is handled in post-processing. If joint velocities were incorporated explicitly in the planning task, it would double the dimensionality of the configuration space while introducing differential constraints. Those differential constraints produce a constraint manifold that has half the dimensionality of the configuration space, and forces child nodes to be dependent on their parent nodes. The former results in slower exploration and the latter is prohibitive for making connection attempts between trees.

II. Constraint Functions

There are four types of constraints which are crucial for humanoid robot manipulation tasks: Balancing constraints, end-effector pose constraints, collision constraints, and joint torque constraints. Some constraints are resolved using recursive hierarchical nullspace projection [3] while others are resolved by simply rejecting configurations that violate them. Typically, a constraint with lower dimensionality than the configuration space is resolved using projection while constraints with wide-open manifolds use rejection.

A. Balance Constraints

A humanoid robot must keep its center of mass above its support polygon in order to maintain quasistatic balance. In some scenarios, the balance constraint could be maintained by rejecting configurations which violate it. However, in a challenging manipulation task, it is unreasonable to simply hope that balanced configurations will be sampled, so a projection operation is needed. We use the center of mass Jacobian method to drive the robot's center of mass into its support polygon.

A more physically meaningful constraint would be to keep the Zero Moment Point (ZMP) above the robot's support polygon [4], but as discussed earlier, the configuration space of the plan only includes joint position and assumes zero instantaneous velocity, so the ZMP constraint would reduce to a simple center of mass constraint. A consequence of the quasistatic assumption is that if a configuration's center of mass is on the edge of the support polygon, the ZMP might be pushed past the support polygon once the velocity of the trajectory is considered. The magnitude of this violation will scale nonlinearly with the magnitude of the velocity, but the violation will approach zero as the magnitude of instantaneous joint velocities approaches zero. In theory, this means that a plan generated with quasistatic assumptions could be infeasible, because it would require the trajectory to come to a full stop and then remain stopped in order to maintain balance. However, in practice, a conservative estimate of the support polygon allows for a margin of error that will allow the robot to continue moving, albeit slowly, even as the center of mass grazes the estimated support polygon. In addition, optimizing the trajectory during post-processing can eliminate this concern altogether.

B. End-effector Constraints

In order to handle end-effector constraints, we employ Task Space Regions [5]. Task Space Regions provide a convenient and generalized way of describing and solving end effector constraints, including articulated constraints. A critical example of how TSRs are used for planning manipulation tasks is for finding kinematically feasible grasp configurations. We assume that any given target object has a set of viable grasp points or grasp regions and then we represent those points or regions as a set of TSRs.

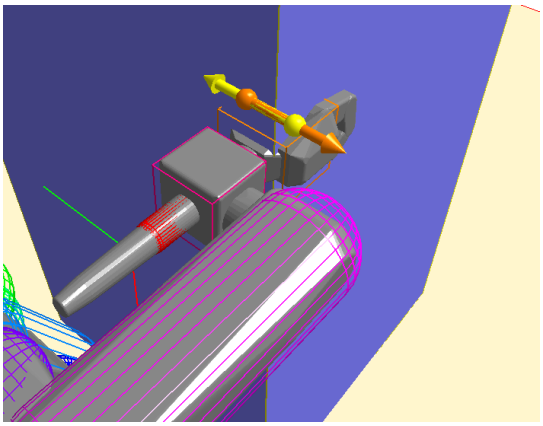
*This work was supported by DARPA award D13AP00047

[1]Michael Grey is a doctoral student in the School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA 30332 mxgrey@gatech.edu

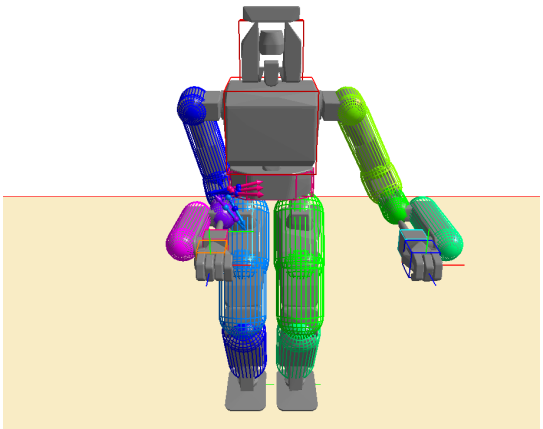
Additionally, we must constrain both feet to remain stationary during a plan. Any relative motion between the feet while both are being used for support would produce dangerous internal forces inside the robot which could result in mechanical or electrical damage. We do not consider the possibility of taking a step during a manipulation motion plan. If foot steps need to be taken, this should be handled by a footstep and walking planner.

C. Collision Constraints

In wide-open environments, collision constraints may be handled by simply rejecting configurations which exhibit collisions. Alternatively, using convex collision geometries allows configurations to be projected out of collision using gradient descent like in [6] and [7]. Escaping collisions using gradient descent can be useful in cluttered environments where the passages between obstacles are thin. Examples of the kind of information that can be derived from convex geometries can be seen in Figure 1. The large dots represent the deepest points of intersection, and the arrows show which way a geometry could be moved in order to escape collision. Applying inverse kinematics methods to that error vector enables the planner to rapidly escape collisions without relying on random sampling.



(a) Collision with an environment object



(b) Self-collision

Fig. 1: Convex collision geometries and the error vectors that can be obtained from them

D. Torque Constraints

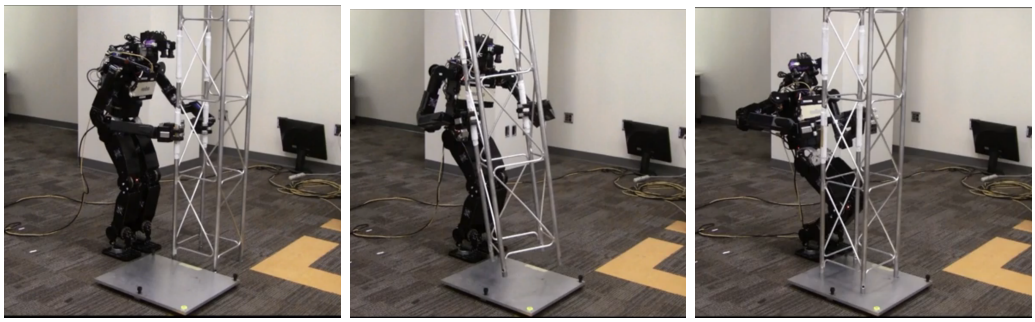
A fourth constraint which may be important to consider (depending on the strength of the robot compared to the task that is being planned) is torque constraints. Since physical motors have torque limits, any configuration whose joint torques exceed those limits must be considered invalid. The probability of randomly sampling a configuration which violates torque limits is usually low enough that simply rejecting invalid configurations (instead of trying to project them) should be suitable. The Jacobian Transpose can be used to compute the static joint torques in the arms for a given configuration, however computing joint torques in the legs may require assumptions about how the weight is distributed between the supporting feet.

E. Context-Sensitive Constraints

A new concept we are exploring is constraints which depend on some contextual parameter that would not normally be included in the configuration space. The first motivating example of this concept is for lifting heavy objects [8] where the constraint manifold is a function of how much of the object's weight is being supported by the robot. Example projections of this manifold can be seen in Figure 3 (these manifolds are only projected so that they can be visualized). We introduce the concept of the Virtual Task Dimension (VTD) to enable the robot to plan the way it transfers the object's weight from the environment to its end effector. This creates a new constraint manifold (seen in Figure 3c) that is able to bridge the two disconnected manifolds (seen in Figure 3b). This new constraint manifold uses all the same constraint functions as the original, except that the balancing constraint function has an additional variable: the fraction (from 0.0 to 1.0) of the target object's weight that the robot is supporting. The numerical value of the VTD is used to represent this new variable. Figure 2 shows the DRC-HUBO robot lifting a metal truss, using a motion that was planned with this approach.

While this is the first use case of the VTD, we believe it should be capable of much broader applications. There are two ideas currently being considered:

1) Planning in a way that exploits friction by conidering the trade off between lifting and dragging. The VTD would represent the same idea as the heavy lifting problem (fraction of object's weight supported by the robot) and the resistive force due to friction can be computed as a function of the VTD. This way, if completely lifting an object would violate torque limits, and completely pushing an object would violate torque limits, the planner could automatically find a compromise between the two and explore those possibilities using the VTD.
2) Plans that can let the robot lean against the environment. The VTD would represent the contact force of an end effector that is pressed against the environment, for example a hand that is pressed against the surface of a table. The feet would still be used as the main



(a) Grabbing the truss (b) Raising the truss (c) Placing the truss

Fig. 2: DRC-Hubo performing a heavy lift task

providers of support, and the support polygon would remain unchanged. However, the additional contact force could be used to help keep the overall ZMP of the robot inside of the support polygon even as the robot reaches over the table to grab an object that would normally be outside of its reach.

III. Finding Start/Goal Configurations

Before CBiRRT can generate a path, it is necessary to determine start/goal configurations. These configurations must satisfy all of the feasibility constraints (i.e. they must be balanced and collision-free) in addition to achieving an objective. The objective can also be represented as a constraint, ideally as a Task Space Region, which describes where the robot should grab or where it should deliver its payload. Because of the high dimensionality of humanoid robot systems and the nonlinearity of the constraint functions, analytical solutions for satisfying all of these constraints simultaneously do not generally exist.

Therefore, in order to solve all the simultaneous constraints, we used stochastic gradient descent. In wide-open, easy environments it might be possible to simply use ordinary gradient descent, but when dealing with cluttered or challenging environments, it is common to get caught in local minima.

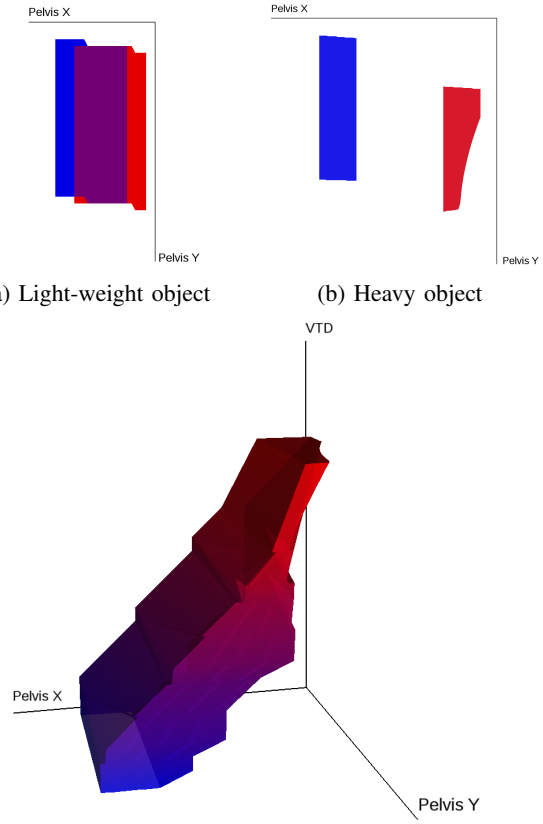
Rather than beginning gradient descent attempts from random configurations, we start them from seeding configurations which are known to be far from singularities or local minima. Samples of the seeding configurations which were used in this project can be seen in Figure 4.

IV. Practical Considerations

Whole body planning is generally difficult due to its high dimensionality and tight constraints. In this section, we will highlight various techniques which help to make it tractable. Some of these techniques are based on the particular use of DRC-Hubo, but may also apply to other humanoid robots.

A. Floating Base Method

Each leg on DRC-Hubo has 6 degrees of freedom, and as mentioned in subsection II-B, we constrain both feet to remain stationary. This means that 12 dimensions of the joint space are fully constrained. If these joints were included



(a) Light-weight object (b) Heavy object

(c) Projection of valid Pelvis X/Y translations along with VTD for the same heavy object as Figure 3b.

Fig. 3: These are projections of two components (Pelvis X and Y) from the 22 dimensional configuration space during a lift. Pelvis X and Y are the X/Y translations of the robot's root Pelvis link. Blue regions are valid while the robot supports none of the object's weight; red regions are valid while the robot supports all of the object's weight. In Figure 3a the purple region represents an overlap of the blue and red regions, indicating that there do exist lifting configurations that are balanced both with and without the object's weight supported. Figure 3b has no such intersection, therefore there is no way to plan a valid path from the blue to the red without introducing the VTD. Figure 3c shows the space of valid configurations for Figure 3b once the VTD is introduced; the planner is able to explore this space freely in order to find a valid path from a point in the blue region to a point in the red region.

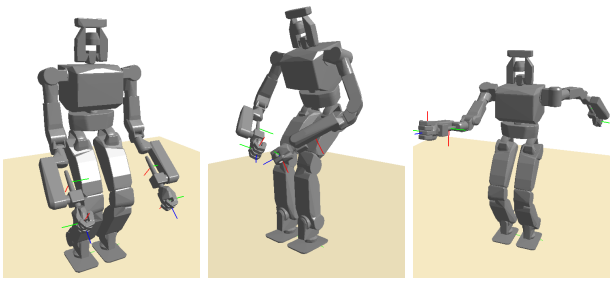


Fig. 4: Examples of useful seeding configurations.

in the configuration space that is used by the planner, an inordinate amount of time would be spent projecting the legs' joint values onto their constraint manifold. Instead, we leave the legs' joint angles out of the configuration space used for planning, and simply use the six degrees of freedom (three translational and three rotational) of the robot's pelvis (root) link. Imagine that the robot's legs are removed and the pelvis could float around and rotate itself freely.

Analytical inverse kinematics is used to ensure that the robot's floating-base poses are feasible. If no valid leg joint configuration exists for a given pelvis pose, the analytical IK is guaranteed to report it. Moreover, even though a 6-DoF analytical IK will usually produce 8 unique joint configurations, the legs on the DRC-Hubo can only ever have up to one valid joint configuration once the legs' joint limits are taken into account. This means that there is no redundancy to account for while generating a plan, and so the planner is *still probabilistically complete* even without explicitly considering these degrees of freedom. We implicitly plan the joint angles of the legs by planning the six degrees of freedom of the pelvis and verifying that there exists a valid set of leg configurations for any given pelvis pose.

B. Task Decomposition

Whenever the constraint manifold of a task has a bottleneck, it is best to decompose the task into two parts where each part takes place on either side of the bottleneck. For example, a pick-and-place task has a bottleneck when it comes to reaching for the object. Since valid grasps are often represented as a finite set of points (or very small regions), the plan needs to pass through a low-dimensional constraint manifold. Instead of making the planner find a path through this bottleneck by exploring, it is far more effective to decompose the task into a picking task followed by a placement task. Then stochastic gradient descent can be used to find an entry point to the bottleneck, and a goal tree can be grown from there.

C. Trajectory Generation

Using CBiRRT generates a path through configuration space, but execution on a robot requires a trajectory which specifies joint values as a function of time. For this we used a time-optimal trajectory generation algorithm by Kunz [9]. This algorithm accepts as input a path as well as maximum speeds and accelerations for each dimension in the configuration space. For the maximum speeds and accelerations, we choose small values to ensure that the robot always maintains quasistatic behavior.

V. Conclusions and Future Work

For humanoid robots to successfully perform physical tasks, there are crucial physical constraints that need to be satisfied. A wealth of techniques from classical mechanics and control theory can be utilized in planners to mathematically guarantee the satisfaction of those constraints. These guarantees are very appealing for the sake of safety and performance. However, there are obvious weaknesses to this approach, in particular that they require experts in mechanics to design and program them. Learning techniques are appealing because they shift the responsibility for success from a human engineer to the robot itself, but learning techniques tend to require a prohibitive amount of failure before positive results can be obtained. On a large-scale humanoid robot, failure is expensive and training is very time consuming. A powerful research topic might be to merge the advantages of constraint satisfaction and learning. Perhaps constraints can be used as a basis for algorithms to learn from, or maybe learning algorithms can be taught to design constraint functions to plan with.

Acknowledgments

This work was funded by the DARPA Young Faculty Award given to Mike Stilman, who tragically passed away before this project started. Nevertheless, we know he would have been proud of our achievements and excited to see where this work leads.

References

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

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

[3] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," in *International Journal of Humanoid Robotics*, 2005, pp. 505–518.

[4] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 2, Sept 2003, pp. 1620–1626 vol.2.

[5] 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.

[6] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *ICRA'09. IEEE International Conference on Robotics and Automation, 2009*. IEEE, 2009, pp. 489–494.

[7] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[8] M. X. Grey, S. Joo, and M. Zucker, "Planning heavy lifts for humanoid robots," in *14th IEEE-RAS International Conference on Humanoid Robots, 2014*, Nov 2014.

[9] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," 2012, p. 209.