More Solutions Means More Problems: Resolving Kinematic Redundancy in Robot Locomotion on Complex Terrain

Brian W. Satzinger[1], Jason I. Reid[2], Max Bajracharya[2], Paul Hebert[2], and Katie Byl[1]

***Abstract*— This paper addresses the open challenge of planning quasi-static walking motions for robots with kinematically redundant limbs. Focusing on RoboSimian, a quadrupedal robot developed by the Jet Propulsion Laboratory (JPL), we develop a practical method for generating statically stable walking motions by pre-computing a reduced dimensional inverse kinematics (IK) lookup table with certain uniqueness and smoothness properties. We then use that lookup table to generate IK solutions at the beginning and end of walking phases (e.g., swing, body shift, etc), and connect these waypoints using the Rapidly exploring Random Tree Connect (RRT-Connect) algorithm [1]. Thus, we avoid arbitrarily choosing an IK solution at the goal (that may turn out to be difficult to reach from the start) by setting this choice through design and use of a task-specific lookup table, which can be analyzed offline. Our approach also introduces a complementary formulation of the RRT-Connect configuration space that addresses contact and closure constraints by using the forward kinematics of one stance leg to determine the body pose while treating additional stance limbs as dependent on the body pose and solving their inverse kinematics with IK table lookup. We demonstrate an implementation of some of this framework on RoboSimian and discuss generalizations and extensions.**

I. Introduction

RoboSimian is a quadruped robot with four identical limbs, which are used both for locomotion and manipulation. This dual purpose forces compromise. Dexterous manipulation requirements favor limbs with redundant degrees of freedom (DOF), while these redundancies can potentially become planning liabilities when walking. Specifically, while setting all 6 DOFs (x, y, z, roll, pitch, yaw) of an end effector is often essential for manipulation, the exact orientation of limb contacts on terrain is often a more open choice, particularly for a robot with four (or more) legs and/or point feet. Extra DOFs give more flexibility in planning, but they also increase the dimensionality of a search for feasible solutions, and the kinematic flexibility of a robot such as RoboSimian also increases the likelihood any potential pose results in collisions, both with the robot and with the external world, dramatically.

Ogura [2] demonstrated walk planning on a biped with 7 DOF legs. The additional DOF allowed yaw motions at the ankle, enabling the knee to be yawed independently of the toe. However, Ogura's planners assume a prescribed knee joint angle, reducing the number of independent DOF to 6 and sidestepping the issue of kinematic redundancies.

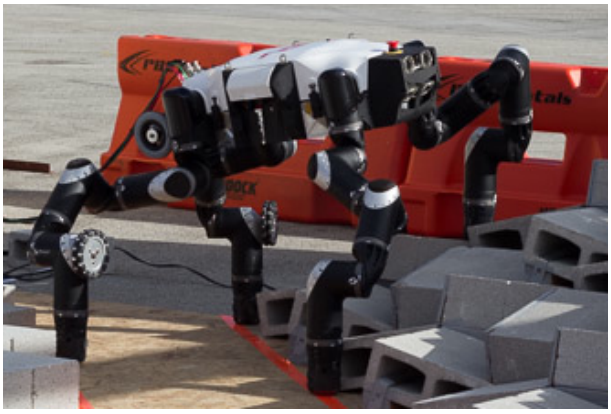


Fig. 1: RoboSimian walking on rough terrain in the DARPA Robotics Challenge (DRC) 2013 Trials.

Kuffner [3] showed a method for determining stable trajectories for a humanoid robot (with 6-DOF limbs) using a variant of RRT-Connect. Single-leg stance poses are generated by randomly sampling the full robot configuration space, fixing one foot to the ground (roughly corresponding to our notion of the dominant limb), and checking for feasibility (static stability, no self collisions, joint torque limits) of the stance. Double-leg stance poses are generated in a similar manner, except that kinematic closure through the other limb is achieved by inverse kinematics from the body to the ground (roughly corresponding to our notion of a dependent limb). Our work extends these concepts in order to address the challenges inherent in robots with more than two limbs or more than 6-DOF per limb.

A body of work also exists around planning walking motions by first searching over a graph of potential stances to find a feasible path to a goal stance, then filling in between these stance transitions as a second planning phase [4] [5] [6]. However, this work has not addressed the complications that arise from redundant limbs when choosing a single robot configuration (i.e., defining all joint angles) to achieve a stance. This is perhaps not surprising because the robots under study were bipeds with 6-DOF legs [7], or a quadruped with 3-DOF legs. Inverse kinematics on these limbs will give a finite set of solutions, and simple heuristic rules (e.g., the knee goes forward) and constraints (e.g., joint limits and self collision) may be sufficient to pick an appropriate solution

[1]Brian W. Satzinger and Katie Byl are with the Robotics Lab in the Dept. of Electrical and Computer Engineering, University of California, Santa Barbara, CA, USA {bsatzinger, katiebyl}@gmail.com

[2]Jason I. Reid, Max Bajracharya, and Paul Hebert are with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA USA, {jreid, maxb, paul.hebert}@jpl.nasa.gov

*The research described in this paper was carried out by UCSB and JPL and is funded by the DARPA DRC program. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

implicitly by process of elimination. However, for robots with 7-DOF per limb (or more), there is no finite list of solutions from which to eliminate, thus an explicit method is required. The choice of method has implications for the broader planning problem because it will (a) affect individual stances, and (b) affect the transitions between stances (e.g., poor start and goal configurations may prevent a stance transition, whereas better start and goal configurations may allow it).

II. Overview of Approach

A. Complications of High DOF Limbs

High DOF limbs and their associated kinematic redundancies introduce several complications into the design of our motion planner. To illustrate these complications with a toy example, first consider a two-link planer limb operating in a planar two dimensional workspace. Walking is a quasi-periodic activity, which may be approximated by a circular reference trajectory for the end effector. For a two-link limb to achieve two end effector coordinates (here, x and y), there will in general be two IK solutions where the elbow bends either way, and a unique IK solution can be generated by consistently picking one of the redundant solutions. Picking one of the IK solutions consistently guarantees that a periodic end effector trajectory will create a periodic joint trajectory. That is, repeated motions (e.g., walking), will not cause the limb configuration to drift over time, as shown in Figure 2.

A two-link IK solution is simple, and efficient solvers such as ikfast [8] exist to solve IK problems analytically on more complex limbs. When limbs have kinematic redundancies, there may be an infinite number of IK solutions. Our challenge is choosing a solution which preserves configuration smoothness over quasi-periodic walking.

One tempting (apparent) solution is to use a Jacobian-based numerical IK solver to update the joint angles as the end effector is moved through small displacements along a trajectory. Pathological situations excluded, this should give a smooth end effector trajectory. However, it is easy to construct examples (Figure 3) where the limb configuration drifts over several cycles, eventually resulting in dramatic changes to the geometry of the limb (e.g., which way knee joints bend). This drift is undesirable since it may lead to collisions, accumulated joint rotations, and otherwise sub-optimal limb states. Indeed, our experience with this approach with RoboSimian has motivated our work here.

We would like to use RRT-Connect to solve for a feasible path to a previously established goal position [1]. With numerical IK solvers, there is no clear way to generate the goal position before generating the end effector trajectory. If we pick one of the analytical IK solutions, how do we know which one to choose in order to avoid configuration changes during the motion?

Another solution is to parameterize joint angles until the redundancies are eliminated. For example, we could simply fix the absolute angle of the last joint on the three-link limb, effectively reducing our three-link limb to a two-link limb standing on a stilt. More complex parameterizations

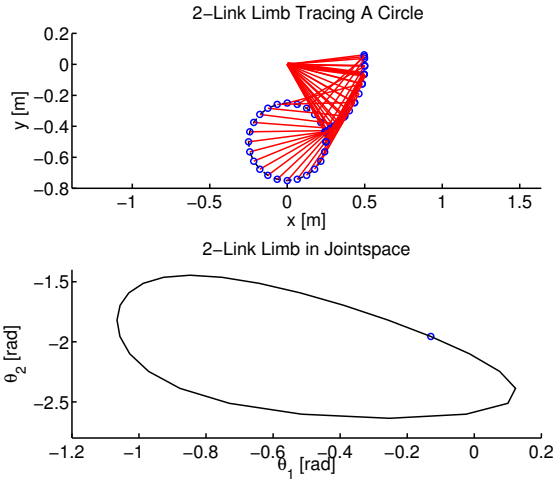


Fig. 2: Periodic joint trajectories naturally develop with a two DOF limb in a 2 dimensional workspace. Top: representation of the limb in task space, showing one loop around the circle. Bottom: the same trajectory in joint space showing that it is a closed loop.

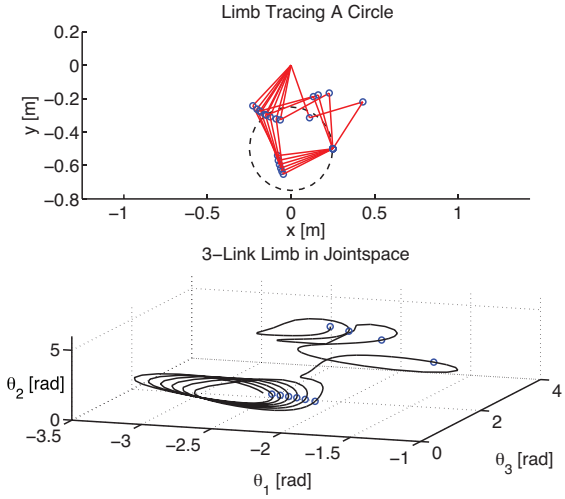


Fig. 3: In contrast to Figure 2, kinematic redundancies in a 3-link arm allow non-periodic solutions in joint space. Top: snapshots of the limb configuration at one point in the trajectory over ten cycles. Bottom: the entire trajectory in joint space. Snapshots drawn above are represented as blue points along the trajectory here.

of the joint angles are possible, and our approach is to do exactly that with a lookup table. This would seem to throw away the advantages of having the additional degrees of freedom, but we will show ways to take advantage of the redundant DOF to generate more dexterous walking. To preview our approach, when walking, we will not require all limbs to use this reduced dimensional IK solution at all times. Sometimes, when permissible, free movement in joint space will be allowed. We will also make use of multiple IK parameterizations for different walking situations and

develop methods for switching limbs between these configurations while walking in an efficient way. Making some limbs dependent on others will also allow us to enforce kinematic closure constraints while reducing the configuration space of our RRT formulation by taking advantage of the uniqueness of our IK solution.

III. Algorithms

We will denote a coordinate transformation from frame a to frame b by C_b^a. Coordinate transforms can be multiplied ($C_c^b C_b^a = C_c^a$) and inverted ($(C_b^a)^{-1} = C_a^b$). We assume a fixed $world$ frame, a $body$ frame attached to the robot body, and foothold frames. We will refer to a foothold (frame) by f, which can be indexed by limb. Therefore, $C_{f_i}^{world}$ gives the location of the i_{th} foothold in the $world$ frame. We will use i as a generic limb index, and sometimes d and s to denote the index of a dominant or swing limb (to be defined) as appropriate. A joint trajectory over time is written $q(n)$, but where n is defined, $q(n)$ is understood to refer to a specific sample. The joint angles of limb i are denoted by q_i, while the joint angles of the entire robot (consisting of appending all of the q_i's) are designated by q. We will also designate optional function parameters (related to the swing leg) in $[brackets]$. Some functions will return a $status$ variable, intended to indicate the $Success$ or $Failure$ of the function call.

We also assume that a forward kinematics function FK is known

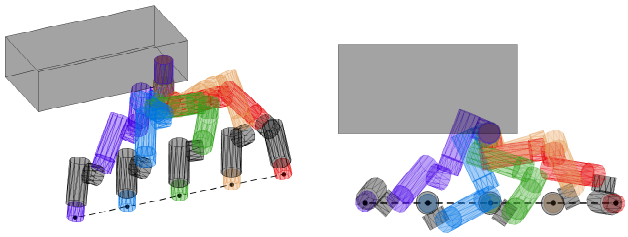
$$C_{f_i}^{body} = FK(i, q_i). \tag{1}$$

$$(q_i, status) = IK\_TABLE(i, C_{f_i}^{body}) \tag{2}$$

We have created an IK lookup table function IK_TABLE (Equation 2) to give us unique IK solutions with certain properties. These properties include that the solutions are smooth in joint space through any closed reachable cycle in the workspace. This may require limiting the reachable workspace to avoid problematic regions. The walk IK table must also be designed so the output poses are geometrically appropriate for walking. Our implementation ignores the orientation given in $C_{f_i}^{body}$, as roll, pitch, and yaw are pre-programmed into the IK solution table so that the orientation is smooth over the feasible workspace. However, this simplification is not required in general. This was done to reduce the IK table lookup to a three-dimensional interpolation of q_i over 8 gridded points defining a cube that encapsulates the position encoded in $C_{f_i}^{body}$.

We have defined an individual limb to be either dominant, swing, or dependent during a motion primitive, and we have defined 3 motion primitives. Table I shows the number of each limb type used during each motion primitive. We will first define the role of the dominant limb. Put simply, the joint angles of the dominant limb determine the body pose.

Figure 5 illustrates how the joints angles of the dominant limb, q_d, and the dominant foothold frame, $C_{f_d}^{world}$, can define the position of the body C_{body}^{world}. In our implementation



(a) Angle View (b) Top View

Fig. 4: Example IK table solutions for a limb on RoboSimian as the end effector moves along a line. Different solutions are identified by color, with one link drawn in black to highlight the differences in its pose as the end effector is moved.

Motion Primitive	Total Limbs	Dominant	Swing	Dependent
Single Step	$N \geq 2$	1	1	$N-2$
Body Shift	$N \geq 2$	1	0	$N-1$
Contact	$N \geq 2$	0	0	N

TABLE I: Limb Modes during Different Walking Primitives. For RoboSimian, $N = 4$

we wanted to allow for ground contact angle to vary, so we augmented q_d with virtual (passive) roll and pitch joints (this must also be accounted for in FK). Lines 1-3 of Algorithm 1 show how to calculate C_{body}^{world} explicitly.

Algorithm 1 Procedure for calculating the full pose (consisting of q and C_{body}^{world}). $C_{f_i}^{world}$ give the foothold locations relative to the world, d and q_d give the index and joint angles of the dominant limb, and optional parameters s and q_s give the index and joint angles of the swing limb (if applicable).

1: **procedure** $FULL\_POSE(C_{f_i}^{world}, [d, q_d], [s, q_s])$
2: $\quad C_{f_d}^{body} \leftarrow FK(d, q_d)$
3: $\quad C_{body}^{world} \leftarrow (C_{f_d}^{body})^{-1} C_{f_d}^{world}$
4: $\quad$ **for** $i = 0$ to $(N_{limbs} - 1), i \neq d, i \neq s$ **do**
5: $\quad\quad C_{f_i}^{body} \leftarrow C_{f_i}^{world} (C_{body}^{world})^{-1}$
6: $\quad\quad (q_i, status) \leftarrow IK\_TABLE(i, C_{f_i}^{body})$
7: $\quad\quad$ **if** $status \neq Success$ **then**
8: $\quad\quad\quad$ **return** $(C_{body}^{world}, q, Failure)$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: $\quad$ **return** $(C_{body}^{world}, q, Success)$
12: **end procedure**

Once the body pose C_{body}^{world} is determined, IK_TABLE can be used to compute the joints q_i of the i_{th} limb (Algorithm 1 lines 5-6). We perform this computation on all limbs except the dominant limb ($i \neq d$) and (if the motion primitive is single step) the swing limb ($i \neq s$). The non-dominant and non-swing limbs are called dependent limbs, because their pose is fully determined by the fixed stance footholds on the ground and the body pose (which is itself fully determined by the dominant limb joints q_d).

When the motion is a single step, one of the limbs is a swing limb. The base of the swing limb is attached to the

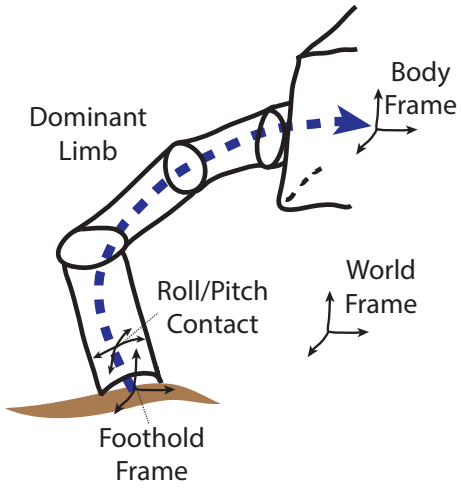


Fig. 5: The dominant limb determines the position of the robot body in the world. A virtual unactuated roll/pitch joint at contact with the ground allows the foot's contact angle with the ground to vary during motions. For robots with four or more limbs, the positions of the dependent limbs will enforce the chosen contact angles.

body, and the dominant limb, body, and swing limb could be conceptualized as one long serial mechanism (Figure 6). The index of the swing limb (s) and the joint angles of the swing limb (q_s) are optionally passed to $FULL\_POSE$. It should be understood that q_s is left unchanged, and is included as one part of the joint vector q (containing all q_i appended together).

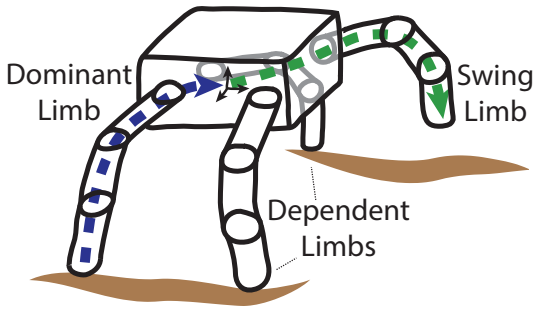


Fig. 6: Illustration of dependent, dominant, and swing limbs. The choice of dominant limb is arbitrary out of the non-swing limbs, although certain choices may give better results than others.

If a feasible solution is found (i.e., "Success"), the output of $FULL\_POSE$ fully specifies a robot configuration, since it returns the global body pose and all joint angles. Therefore, the arguments to $FULL\_POSE$ can be used to define a configuration space for an RRT-Connect solver. In particular, $C_{f_i}^{world}$, d, and s provide context for the problem by defining the foothold locations, and which limbs are dominant and stance. For a body shift motion, q_d is the configuration space, while for a single step, the concatenation $(q_d\, q_s)$ is the configuration space. The number of dimensions in the configuration space for the single step and body shift motions are given in Table II. Note that the number of dependent limbs does not affect the number of dimensions. For RoboSimian, the single step configuration space has 16 dimensions, and the body shift configuration space has 9 dimensions ($size(q_d) = 9$, and $size(q_s) = 7$).

Motion Primitive	Actuated DOF per Limb	Passive DOF at Dom. Contact	RRT Dimensions
Single Step	K	2	$2K+2$
Body Shift	K	2	$K+2$

TABLE II: RRT Configuration Space Size During Different Walking Primitives

Algorithm 2 Procedure for determining the RRT-Connect configuration $(q_d, [q_s])$ from footholds $C_{f_i}^{world}$ and a body pose C_{body}^{world}.

procedure $CONFIG(C_{f_i}^{world}, C_{body}^{world}, d, [s])$
  for $i = 0$ to $(N_{limbs} - 1)$ **do**
    $C_{f_i}^{body} \leftarrow C_{f_i}^{world}(C_{body}^{world})^{-1}$
    $(q_i, status) \leftarrow IK\_TABLE(i, C_{f_i}^{body})$
    if $status \neq Success$ **then**
      return $(q_d, [q_s,]\, Failure)$
    end if
  end for
  return $(q_d, [q_s,]\, Success)$
end procedure

RRT-Connect requires pre-computing the goal location before performing the search. Algorithm 2 provides an algorithm ($CONFIG$) for doing so, by using IK_TABLE to determine the limb joint angles q_d and q_s given footholds, a body pose, and the dominant and swing leg indices. Using IK_TABLE to determine the goal positions takes advantage of the uniqueness and smoothness properties designed into the table and prevents configuration drift over subsequent steps. When the motion is a step, the destination foothold ($C_{swing\_dest}^{world}$) is passed as a parameter. We do not directly address how to choose the next foothold, but existing foothold planning methods are applicable here [9], [10].

Algorithm refalg:solve shows the function ($SOLVE$) we use to wrap the RRT-Connect ($RRTC$) planner and produce a goal trajectory. It requires specifying which limb is the dominant limb (and swing limb). However, any non-swing limb can in principle be the dominant limb and our experience suggests that the choice of dominant limb makes a significant difference in the difficulty of a given planning problem. Because $SOLVE$ is not very time-consuming to run, we simply try all possible choices for d and pick the solution with the shortest execution time (respecting actuator velocity and acceleration limits). $PLAN$ accomplishes this by wrapping $SOLVE$. In turn, $PLAN$ takes the goal body position $C_{robot,goal}^{world}$ as a parameter. We determine this body pose (externally to $PLAN$) with a heuristic method based on the foothold locations.

We also leave implicit that $RRTC$ will have access to the problem context consisting of $C_{f_i}^{world}$, d, and s. During

$RRTC$ planning, each node (state) that is added to the tree is verified to be kinematically feasibile (as determined by $FULL\_POSE$), statically stable (the center of mass is above the convex hull of the stance footholds), and collision free (checked using Bullet Physics Library [11]).

Algorithm 3

procedure $SOLVE(C^{world}_{f_i}, C^{world}_{robot,start}, C^{world}_{robot,goal}, d, [s, C^{world}_{swing\_dest}])$
  $C^{world}_{f_{goal,i}} \leftarrow C^{world}_{f_{start,i}}$
  if planning swing (s exists) **then**
    $C^{world}_{f_{goal,s}} \leftarrow C^{world}_{swing\_dest}$
  end if
  $start \leftarrow (C^{world}_{f_{start,i}}, C^{world}_{robot,start}, d, [s])$
  $goal \leftarrow (C^{world}_{f_{goal,i}}, C^{world}_{robot,goal}, d, [s])$
  $(q_{d,start}, [q_{s,start}]) \leftarrow CONFIG(start)$
  $(q_{d,goal}, [q_{s,goal}]) \leftarrow CONFIG(goal)$
  $(q_d(n), [d_s(n)], status) \leftarrow RRTC((q_{d,start}, [q_{s,start}]), (q_{d,goal}, [q_{s,goal}]))$
  if $status \neq Success$ **then**
    return $(Null, [Null,] Failure)$
  end if
  for $n = 0$ to $length(q_d)$ **do**
    $(C^{world}_{robot}(n), q(n)) \leftarrow FULL\_POSE(C^{world}_{f_i}, d, q_d(n), [s, q_s(n)])$
  end for
  return $(C^{world}_{robot}(n), [q(n),] Success)$
end procedure

During the Contact motion primitive, all limbs are considered dependent. This is a special type of motion used to make or break contact with the walking surface. A contact primitive is used before and after a Single Step to move the swing foot to and from a standoff location a small distance above the foothold. This removes the swing foot from collision with the terrain model. Here, all limbs, including the swing limb, are considered dependent because their configurations are restricted to IK-table solutions. The trajectory of the dependent swing limb's end effector is simply a Cartesian interpolation between the foothold and the standoff position. During a contact behavior after a step, when the foot is approaching the ground, uncertainty in the ground position can be accounted for if there are force sensors in the foot. A contact trajectory is generated that penetrates through the expected location of the ground and the trajectory is halted when the robot detects contact with the ground. Since all limb positions are, by design, IK table solutions during a Contact trajectory, the trajectory can be terminated at any time. The resulting final pose will therefore always be valid as a starting pose for the next primitive, even if there was an error in the expected position of the ground (the error must be within the bounds of the standoff and penetration distances), as long as the end effector does not exit the reachable workspace of the IK table solution set.

Of course, not all configurations are admissible as part of a solution. The initial, final, and intermediate solutions must be statically stable, kinematically feasible, and collision free (for both self-collisions and the environment). We determine static stability by projecting the center of mass onto the ground plane and checking that it is within a support polygon, with an inscribed safety margin. To speed up this calculation, a table is populated with limb center of mass with respect to the robot for all IK solutions at run-time. This reduces the center of mass calculation for each dependent limb to a table lookup (then transform back into world frame). In addition to static stability, other more sophisticated stability constraint criteria could be substituted to allow for more general contacts [12]. Kinematic feasibility includes respecting joint limits on the dominant and swing limbs, and enforcing that the dependent limbs are in the reachable workspace of the IK table. Limits can also be placed on the passive roll-pitch DOF at the foothold of the dominant limb, as appropriate given the geometry of the rolling surface on the robot's foot and the properties of the walking surface.

Algorithm 4

procedure $PLAN(C^{world}_{f_i}, C^{world}_{robot,start}, C^{world}_{robot,goal}, [s, C^{world}_{swing\_dest}])$
  for $d = 0$ to $(N_{limbs} - 1), d \neq s$ **do**
    $exec\_time_d \leftarrow \infty$
    $(C^{world}_{robot}(n), q(n), status) \leftarrow SOLVE(C^{world}_{f_i}, C^{world}_{robot,start}, C^{world}_{robot,goal}, d, [s, C^{world}_{swing\_dest}])$
    if $status \neq Failure$ **then**
      $solution_d \leftarrow (C^{world}_{robot}(n), q(n))$
      $exec\_time_d \leftarrow EXEC\_TIME(solution_d)$
    end if
  end for
  if any solutions were found **then**
    $d_{best} \leftarrow argmin(exec\_time)$
    return $(solution_{d_{best}}, Success)$
  else
    return $(Null, Failure)$
  end if
end procedure

We detect collisions with Bullet using simplified bounding boxes to represent both the robot and the environment. We selectively filter out the collision geometry of the dominant and dependent end effectors because they will naturally be in contact with the ground or other obstacles. When there is a swing limb, the collision geometry of the end effector is not filtered out so that it is prevented from colliding with the ground during swing. Therefore, the start and goal poses of a single step motion must already have the swing limb decollided. The Contact motion primitive, as described above, is used to accomplish this.

A. Limb Configuration Changes

In our definitions, we have constrained the initial and final positions of the swing and dominant limbs to always be IK table solutions. However, at intermediate points in the trajectory the RRT-Connect solver is free to choose any

joint solution, subject to joint limits, stability, and collision constraints. We have emphatically not limited ourselves to a single IK table. Indeed, for our robot there are multiple choices of reasonable IK tables with tradeoffs between the shape of the reachable workspace and the space occupied by the robot and limbs during walking. For example, walking through a narrow door requires the legs to be tucked in toward the body to allow the robot to pass through the width of the door frame. However, there are other walking tables (without the legs tucked in) that are better suited to walking in less constrained environments.

A limb can be transitioned from one IK table to another IK table when it is either a dominant or a swing limb because the initial and final poses can be chosen from different IK tables (when $CONFIG$ is called). The limb undergoing this transition may pass near singularities or make large changes in its configuration. When it is the swing limb, the end effector traces a (potentially labyrinthine) path through free space to accomplish the transition. The RRT-Connect solver, being aware of collision constraints, will avoid collisions of the robot with itself or the environment. When a configuration change is planned for the dominant limb, however, the transition could create a large excursion of the body position because the end effector must remain in contact with the ground. This is correspondingly more likely to inhibit feasibility of any full pose solutions on the basis of kinematic feasibility of the dependent limbs, collision constraints, and stability constraints. For this reason, it is generally preferable to transition a limb between IK tables only during swing.

B. Recovering from Plan Failures

We use RRT-Connect to find a path between the initial and final configurations of the robot. RRT-Connect may fail to find a plan within time and memory constraints, or it may simply be that no full pose solution is possible. Although it is impossible to know to which category any individual failure belongs, re-running the planner with a different random seed, or moving the body to a different physical starting place can work around many failures, from our planning experience with RoboSimian. Other methods to work around failures include changing the initial or final configuration of the robot, such as by changing the body pose. The final configuration can be easily changed by simply generating a different final pose. However, the initial configuration will often correspond to the live pose of the physical robot. Therefore, changing the initial configuration would mean, for example, adding another small body shift (or other motion) before re-planning the motion that failed. These shifts are currently determined by the human operator, but automating this process is part of our future work.

IV. Conclusion

To date, we have implemented a subset of this framework and tested it in software simulation and on hardware. We have designed IK tables that we have nicknamed ”fast”, ”dexterous”, and ”narrow”. The ”fast” table is designed to prioritize forward walking speed across relatively flat terrain, the ”dexterous” table prioritizes walking on uneven terrain, and the ”narrow” table brings the legs under the body to allow passing through doors.

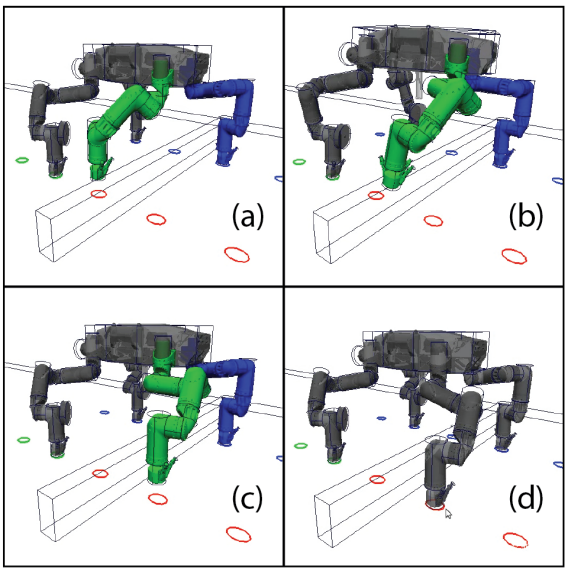


Fig. 7: Visualization of a single step plan over a perceived obstacle and a contact behavior. (a) initial pose including a standoff distance (b) during swing, passing over obstacle (c) at end of swing including a standoff distance (d) final pose after contact motion. Dominant limb is colored blue, swing limb is colored green, and dependent limbs are colored grey. Frame (d) shows the end pose of a contact motion, where all limbs are dependent.

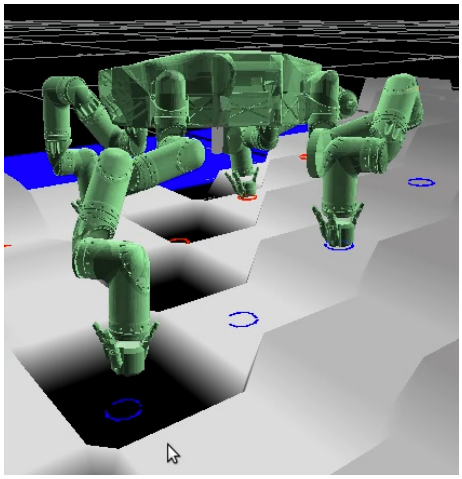


Fig. 8: Screenshot of a kinematic simulation showing walking across DRC cinderblock terrain.

Figure 7 shows the single step and contact planners operating with an obstacle, while Figure 8 shows a still frame from an animation of RoboSimian crossing DRC-style terrain using our approach. Due to limited hardware availability, we have only tested a limited amount of this

framework (roughly corresponding to the capabilities shown in Figure 7) on hardware to date, as shown by a video frame in Figure 9. We anticipate future work will quantify more rigorously the performance of our algorithms relative to existing approaches. Qualitatively, we see planning times on the order of 1 second or less for typical motions. We are able to plan step lengths of up to 88cm on flat ground in simulation using IK tables, whereas RoboSimian's previous walking algorithms were limited to about 60cm length steps.

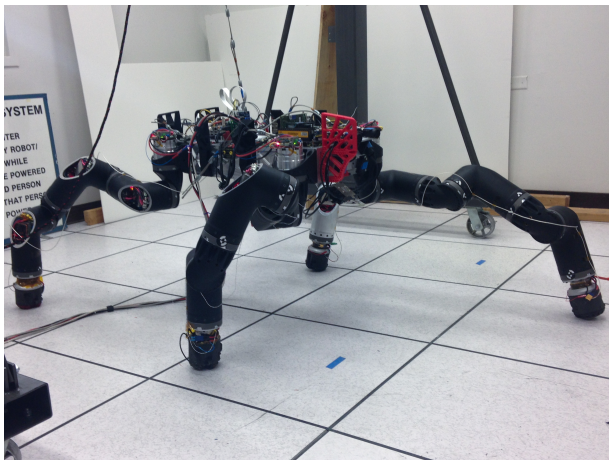


Fig. 9: Testing the planner on RoboSimian hardware. We were able to extend the reachable workspace of our IK table by pitching the feet near the edge of the workspace. This enables longer steps than methods that constrain the foot to be normal to the ground.

V. Future Work

We anticipate further research to explore several promising extensions to this framework. This includes a more thorough stability check, using multiple contact points on the same limb, and walking with multiple simultaneous swing limbs.

Our current stability criteria is that the robot's center of mass must be located above the support polygon. However, this method is known to fail when the footholds are not flat. We plan to replace our stability criteria with one from [12] to handle this properly. We are also ignoring dynamic stability because in practice RoboSimian's actuators are not fast enough to destabilize the robot. We simply generate a quasi-statically stable trajectory, and adjust the playback speed so that all actuators are within their velocity and acceleration limits. Further work will properly account for dynamic stability by reducing the playback speed as necessary (it is always possible to do so because our trajectories are quasi-static).

In general, there may be reasonable contact points on a limb besides the end effector, such as a knee. We plan to allow for their use in the existing IK table framework by generating IK tables and forward kinematics functions to the alternate contact points. Transitions between different contact points are handled as limb configuration changes as described in Section III-A. We expect this to increase RoboSimian's locomotion performance by providing additional foothold opportunities and by freeing an end effector (e.g., to carry an object while using an elbow as a contact support on terrain) while walking.

We have focused on statically stable quadrupedal walking, which requires three stance limbs and one swing limb. However, our method is applicable to robots with more than four limbs as well. As long as three stance limbs are maintained, additional limbs can either remain in contact with the ground as dependent limbs, or become an additional swing limb. Each additional swing limb with K DOF will add K DOF to the RRT configuration space, while each additional dependent limb does not affect the size of the configuration space (although it may affect feasibility of points within that space). For example, a hypothetical hexapod version of RoboSimian could take a triple step (3 swing limbs, 1 dominant limb, and 2 dependent limbs). The configuration space would require 9 DOF for the dominant limb, and 7 DOF for each of three swing limbs, resulting in 30 DOF. Adding sufficiently many DOF will, of course, affect the tractability of the planning problem.

Acknowledgment

The authors would also like to thank the entire RoboSimian team for their efforts in designing the robotic system hardware and software.

References

[1] J. Kuffner and S. 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.

[2] Y. Ogura, T. Kataoka, K. Shimomura, H. ok Lim, and A. Takanishi, "A novel method of biped walking pattern generation with predetermined knee joint motion," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, 2004, pp. 2831–2836 vol.3.

[3] J. Kuffner, JamesJ., S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-stable motion planning for humanoid robots," *Autonomous Robots*, vol. 12, no. 1, pp. 105–118, 2002.

[4] K. Bouyarmane and K. A., "Humanoid robot locomotion and manipulation step planning," *Advanced Robotics (Int. J. of the Robotics Society of Japan), Special Issue on the Cutting Edge of Robotics in Japan 2012*, vol. 26, no. 10, pp. 1099–1126, July 2012.

[5] K. Hauser, T. Bretl, and J.-C. Latombe, "Non-gaited humanoid locomotion planning," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*. IEEE, 2005, pp. 7–12.

[6] T. W. Bretl, "Multi-step motion planning: Application to free-climbing robots," Ph.D. dissertation, Citeseer, 2005.

[7] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot hrp-2," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 2, 2004, pp. 1083–1090 Vol.2.

[8] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, August 2010.

[9] K. Byl, "Metastable legged-robot locomotion," Ph.D. dissertation, MIT, 2008.

[10] P. Vernaza, M. Likhachev, S. Bhattacharya, S. Chitta, A. Kushleyev, and D. D. Lee, "Search-based planning for a legged robot over rough terrain," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2380–2387.

[11] E. Coumans. Bullet physics library. [Online]. Available: https://code.google.com/p/bullet/

[12] T. Bretl and S. Lall, "Testing static equilibrium for legged robots," *Robotics, IEEE Transactions on*, vol. 24, no. 4, pp. 794–807, 2008.