2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

A Framework for Extreme Locomotion Planning

Christopher M. Dellin and Siddhartha S. Srinivasa
The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
{cdellin,siddh}@cmu.edu

***Abstract*— A person practicing parkour is an incredible display of intelligent planning; he must reason carefully about his velocity and contact placement far into the future in order to locomote quickly through an environment. We seek to develop planners that will enable robotic systems to replicate this performance. An ideal planner can learn from examples and formulate feasible full-body plans to traverse a new environment. The proposed approach uses *momentum equivalence* to reduce the full-body system into a simplified one. Low-dimensional trajectory primitives are then composed by a sampling planner called *Sampled Composition A** to produce candidate solutions that are adjusted by a trajectory optimizer and mapped to a full-body robot. Using primitives collected from a variety of sources, this technique is able to produce solutions to an assortment of simulated locomotion problems.**

I. Introduction

A human practicing parkour (Figure 1), also known as "freerunning" or "the art of movement,"[1] is an impressive example of extreme locomotion. The practitioner navigates through complex 3D terrain at high speeds, and is forced to reason about his body and contacts with his environment through both traditional foot and hand holds and other forms of contact (e.g. rolling). In order to clear obstacles and maintain control, he must consider the future; often, he examines the environment in detail beforehand to formulate an explicit plan. Parkour requires planning for a high-dimensional hybrid dynamic system where successful plans involve irregular environment contact as well as high-velocity maneuvers.

We seek to develop planners that will enable legged robotic systems (e.g. humanoids and quadrupeds) to solve these extreme locomotion problems. A planner is necessary because a purely reactive controller is unable to reason about the future; its choice of foot placement must account for obstacles several steps into the future. Such a planner must be able to formulate high-velocity plans. Planners which assume quasistatic transitions are not capable of producing suitable plans for most extreme locomotion problems. It should also have the capacity to easily learn new techniques to broaden the class of problems it can solve.

Planning for extreme locomotion in robotics presents a number of challenges. Motion planning for complex robots is a particularly high-dimensional problem (typical humanoid robots have more than 30 actuated degrees of freedom); this precludes naïve implementations of common planners (e.g. RRTs, graph search methods). These systems are underactuated (indeed, often in full flight), complicating the control problem. They are also hybrid systems, undergoing both discrete and continuous dynamics – planners must be able to reason about phase changes while making and breaking environment contact. Their motion is also constrained by dynamic contact force constraints, closed-loop kinematic constraints, and actuator position and force limits. A successful planner must carefully navigate all of these constraints.

[1]See http://en.wikipedia.org/wiki/Parkour.

Fig. 1. Two men practice parkour by maintaining balance and quickly traversing obstacles using hands and feet [1].

A. Related Work

Variations of the extreme locomotion problem have been addressed by both the bipedal walking and the animation communities. Common to all these approaches is a simplification of the large state space.

Developed more than forty years ago, the Zero Moment Point (ZMP) stability criterion [2] allowed a concise representation of an important constraint for bipedal walking. Early methods simplified the planning problem in three ways: (a) constraining contact to a horizontal plane, (b) enforcing a particular gait (i.e. contact order), and (c) enforcing statically-stable endpoints of dynamically-balanced segments [3]. These segments could then be composed for locomotion. The planners for most modern humanoid robots are direct descendants of this approach. For example, Chestnutt, et. al. [4] used a heuristic-based planner to plan paths for the Honda ASIMO while avoiding moving obstacles.

Recent graph-search methods have allowed researchers to relax the horizontal-plane constraint and consider rough terrain. In the LittleDog project, researchers [5], [6], [7] developed more capable planners for gaited rough-terrain planning for quadrupeds using heuristic-based algorithms. These algorithms rely on a fixed gait to reduce the search dimensionality.

Many researchers in the computer graphics community have considered physical feasibility when generating the motion of animated characters. Mordatch et. al. [8] used an optimization approach to solve a Spring-Loaded Inverted

978-1-4673-1405-3/12/$31.00 ©2012 IEEE

Pendulum (SLIP) model with a regular gait to achieve walking and running motion.

In order to enable non-gaited planning, it was necessary to further strengthen constraint (c), and enforce static stability at *all times* during locomotion. Kuffner [9] used sampling-based algorithms for flat-plane non-gaited humanoid motion planning.

While the ZMP constraint is important to consider for horizontal-plane locomotion, frictional constraints are also important, especially when extending non-gaited planning to rough terrain. Therefore, the "linear wrench feasibility" criterion was developed [10]. Bretl [11] and Hauser [12] used this more general criterion to create statically-stable primitives for climbing robots using a non-gaited PRM planner [13]. Recently, Collette et. al. [14], [15] have developed a dynamic balance regulation framework using the linear wrench feasibility criterion.

In the field of computer graphics, researchers use recorded motion data, organized into a motion graph or database. Lee et. al. [16] showed a technique for preprocessing motion data for realtime control of an animated character. Choi et. al. [17] used Probabalistic Roadmaps (PRMs) to find sequences of motion clips to navigate through an environment. Reitsma et. al. [18] directly unrolled motion graphs into an environment in order to find suitable paths. Safonova et. al. [19] showed an approach for interpolating motion graphs to generate natural-looking, optimal motions to follow a rough user-designed path. Recent work in physically-based motion clip retargeting (e.g. [20]) may allow for robust control of several useful dynamic motions (e.g. rolling).

Researchers have also designed and built mechanisms that perform dynamic locomotion reactively without planners. Degani [21] developed several dynamic hopping and climbing mechanisms with minimal actuation. The RHex robot [22] exhibits dynamic running behavior over rough terrain using six spinning compliant legs.

B. Extreme Locomotion Planning

The techniques developed in related work are inadequate to solve many extreme locomotion problems. Many planners impose one or more constraints that are too restrictive (e.g. flat-plane, quasistatic stability, or fixed gait). Other techniques produce dynamic motion using reactive controllers which are ill-suited to difficult problems which require planning.

We address these shortcomings with our framework for extreme locomotion planning, described in Section II. Our framework is founded upon three key insights: (a) dimensionality reduction by *momentum equivalence*, (b) learned lower-dimensional trajectory primitives, and (c) hierarchical refinement via optimization.

Our framework is able to produce gaitless, dynamic plans for complex legged robots using learned trajectory primitives. The planner can transition smoothly at high velocity between different behaviors. The resulting system is able to solve difficult extreme locomotion problems.

We are excited to be pushing the limits of locomotion planning. However, there is a long way to go. Our current implementation makes a number of assumptions. For simplicity, we assume a static environment with previously-identified contact locations. We do not consider plan controllability, robustness, or collision avoidance, although these avenues are discussed briefly in Section VIII. Notably, we assume perfect state estimation, which allows for ground speed matching and absolves us from an impact contact model. However, these limitations are not inherent in the framework; future work can incorporate the necessary extensions.

II. A Framework for Extreme Locomotion Planning

Motivated by the form of the dominating constraints inherent in the problem, we propose a simplification method and corresponding hierarchical planning framework (see Section II) that fundamentally plans for the momentum of the robot using low-dimensional trajectory primitives. This section defines the extreme locomotion problem and discusses the framework. Subsequent sections (III–VII) describe each component in more detail.

A. Problem Statement

Consider a floating-base rigid-body robot model $\mathcal{R}$ with n internal degrees of freedom. We represent the state of the robot as $\mathbf{x} = [\mathbf{q}; \dot{\mathbf{q}}]$, with $\mathbf{q}$ the $(n+6)$-dimensional vector consisting of the pose (i.e. Cartesian position and orientation) of a root link and the n generalized internal joint coordinates, and $\dot{\mathbf{q}}$ their velocities. The system's evolution is controlled by applying n internal joint torques τ, and through contact forces from the environment.

Each of $i \in \{1 \dots n_c\}$ body-fixed contacts is described by a rigidly-attached frame; the function $\mathbf{y}_i(\mathbf{x})$ gives its pose, and $\mathbf{v}_i(\mathbf{x})$ gives its spatial (linear and angular) velocity. Each contact has an associated set Y_i of potential world-fixed positions pre-selected in the environment. If $\mathbf{y}_i(\mathbf{x}) \in Y_i$ with $\mathbf{v}_i(\mathbf{x}) = \mathbf{0}$, then the robot is in contact through c_i, and a spatial force $\mathbf{f}_i$ can be transmitted from the environment to the appropriate link. For example, the pose $\mathbf{y}_f$ of a flat foot may be described by a frame attached to its center, and Y_i would include all planar positions and orientations on a floor.

The dynamics of the system can be expressed as

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \mathbf{S}\,\tau + \sum_{i=1}^{n_c} \mathbf{J}_i^T(\mathbf{q})\,\mathbf{f}_i, \qquad (1)$$

with $\mathbf{M}(\mathbf{q})$ the mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ the vector of Coriolis and centripetal terms, $\mathbf{g}(\mathbf{q})$ the vector of gravity terms, $\mathbf{S} = [\mathbf{0}_{6\times n}; \mathbf{E}_n]$ the actuated-joint selection matrix, τ the vector of joint torques, and $\mathbf{J}_i(\mathbf{q})$ the ith contact Jacobian.

The system's evolution is subject to a number of constraints, which we classify as either *internal* or *external*. Internal constraints include joint/actuator position, velocity, and force/torque limits. External constraints consist of limits on contact force, and account for ZMP and (potentially

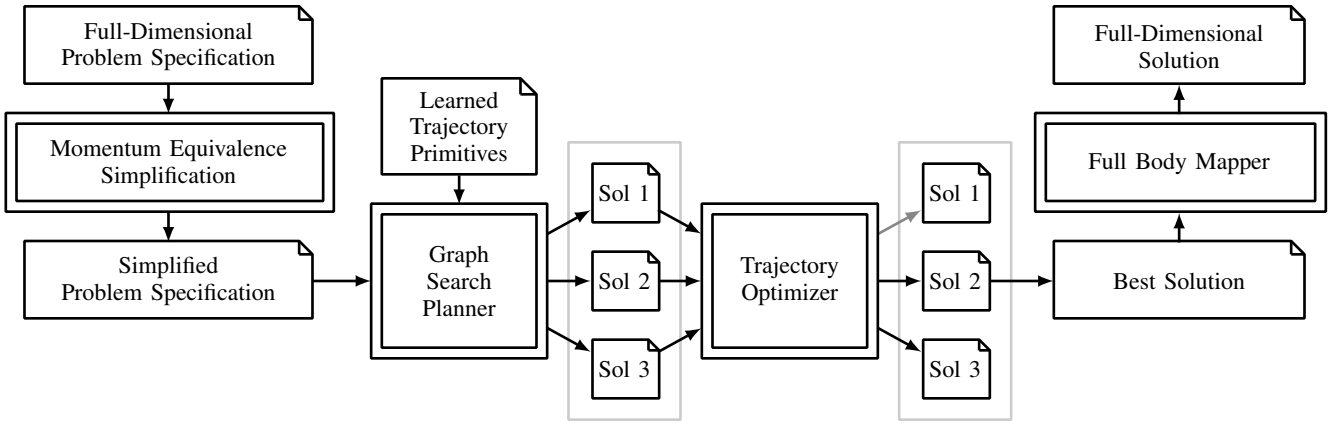


Fig. 2. The hierarchical planning framework proposed here reduces the dimensionality of the motion planning problem to consider the momentum of the robot. The SCA* algorithm produces a set of candidate solutions, which are then optimized and mapped back to the full-body robot.

unilateral) frictional constraints. These constraints can be written as a set of inequalities linear in the contact force:

$$\begin{aligned} \mathbf{B}_i\,\mathbf{f}_i(t) \geq \mathbf{b}_i \quad &\text{if } \mathbf{y}_i(t) \in Y_i, \mathbf{v}_i(t) = \mathbf{0} \\ \mathbf{f}_i(t) = \mathbf{0} \quad &\text{otherwise.} \end{aligned} \tag{2}$$

For example, the force constraints on a foot making contact with a frictional surface can be approximated using the laws of Coulomb friction as an n-sided polyhedral convex friction cone. With the spatial force $\mathbf{f}_i$ expressed in the local contact frame at $\mathbf{y}_f$ during contact, $\mathbf{B}_i$ is a constant basis for the cone, and $\mathbf{b}_i$ is the zero vector.

Fundamentally, we seek a control trajectory $\tau(t)$ for robot $\mathcal{R}$, and the associated state and contact force trajectories $\mathbf{x}(t)$ and $\mathbf{f}_i(t)$, to get from a starting state $\mathbf{x}(t_s) = \mathbf{x}_s$ to a goal state $\mathbf{x}(t_g) = \mathbf{x}_g$, subject to both the internal and external (2) constraints.

B. Framework Overview

Figure 2 presents an overview of the framework. Each component is briefly described here.

We postulate that in the case of extreme locomotion problems, *system momentum* is of primary importance. When planning for high-velocity maneuvers, the practitioner's precise joint trajectories are largely unimportant. Instead, the gross motion of the body, along with the evolution of the contact state, is more relavent. Consequently, the external contact constraints (2) dominate the planning problem. We choose a dimensionality reduction that fully preserves the relevant aspects of $\mathcal{R}$ with regard to these constraints.

Since system momentum is the invariant in the simplification, we call the property *momentum equivalence*. The simplified system $\hat{\mathcal{R}}$ has equivalent linear and angular momentum to the full system. See Section III for details.

Once in the simplified space, we use a library of learned trajectory primitives to find candidate trajectories for the simplified robot $\hat{\mathcal{R}}$. Learned trajectory primitives are described in Section IV. In our implementation, we use a novel heuristic-based algorithm called *Sampled Composition A** (Section V).

Because the library consists of single examples, the resulting trajectories may not be feasible. Therefore, a trajectory optimizer is used to modify the parameters of each potential solution. Plans that are successfully optimized satisfy the external contact constraints (2) and internal constraint approximations. Note that this plan consists only of a center-of-mass trajectory and the locations and timings of each environment contact.

Once a plan is found, a second algorithm (described in Section VII) attempts to find a full-body trajectory for $\mathcal{R}$ that satisfies the full-body internal constraints.

C. Framework Limitations

As with any planning hierarchy, the framework cannot guarantee that a given low-dimensional plan will be feasible when passed to subsequent layers. We therefore produce a set of different candidate trajectories in case some fail.

III. Simplification by Momentum Equivalence

Due to the high dimensionality of the full-body locomotion planning problem described above, we must make significant simplifications in order to successfully find solutions. Unfortunately, the existing simplifications (e.g. enforced gaits or quasistatic stability) do not allow for high-velocity planning using multiple contacts.

In Section II, we differentiated between *internal* and *external* constraints. We further postulated that for extreme locomotion, external constraints dominate the planning problem. Therefore, it is important that our simplification preserves those constraints exactly. We use the concept of *momentum equivalence* as that simplification. Using this technique, we can greatly reduce the dimensionality of the problem.

A. Contact Constraints and System Momentum

Recall the problem statement (Section II-A). In addition to the general governing equation (1), the robot $\mathcal{R}$ as a system also obeys Newton's second law:

$$\dot{\mathbf{h}} = \mathbf{f}_g + \sum_{i=1}^{n_c} \mathbf{f}_i \tag{3}$$

That is, the time-derivative of the robot's spatial (linear and angular) momentum $\mathbf{h}$ is a function only of the gravity force

$\mathbf{f}_g$ and the external forces $\mathbf{f}_i$ exerted at the contact points. It is independent of its internal joint trajectories and torques.

B. System Simplification by Momentum Equivalence

We choose to approximate $\mathcal{R}$ with a simplified robot $\hat{\mathcal{R}}$ consisting of a single rigid body with massless floating contacts $\hat{c}_i$. The simplified state $\hat{\mathbf{x}}$ is then the pose and spatial velocity of this body, along with the contact state. Because the contacts are massless, their dynamics are ignored.

The rigid body $\hat{\mathcal{R}}$ is chosen to have the same total mass and moment of inertia as does $\mathcal{R}$ in a default configuration. The problem's full-body start state $\mathbf{x}_s$ is simplified to the simplified state $\hat{\mathbf{x}}_s$ as follows (the goal state $\mathbf{x}_g$ is simplified in the same way). The linear position and velocity of the center of mass of the rigid body $\hat{\mathcal{R}}$ is chosen to match that of $\mathcal{R}$. The orientation of the rigid body is chosen to match that of the body of $\mathcal{R}$ that is selected as its orientation analogue (e.g. a pelvis or torso). Its angular velocity is selected such that the total angular momentum of the two systems is identical.

In this way, under the same external forces,

$$\mathbf{h}_{\mathcal{R}}(t) = \mathbf{h}_{\hat{\mathcal{R}}}(t), \tag{4}$$

and the centers of mass of $\mathcal{R}$ and $\hat{\mathcal{R}}$ coincide. Note that the orientation of $\hat{\mathcal{R}}$ will not implicitly match the orientation of any body in $\mathcal{R}$. When mapping back to the full robot (described in Section VII), care must be taken to design a trajectory of the internal joints of $\mathcal{R}$ to maintain the desired orientation of the selected orientation analogue body.

Since the kinematic structure of $\mathcal{R}$ is no longer present, position-based approximations are used to keep the contacts c_i within a reachability region relative to the rigid body $\hat{\mathcal{R}}$. For example,

$$||\mathbf{y}_i(t) - \mathbf{y}_{com}(t)|| \leq d_i. \tag{5}$$

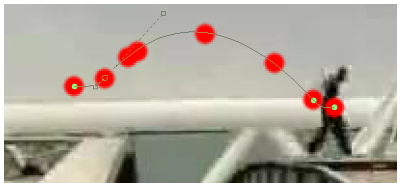
Similarly, since $\hat{\mathcal{R}}$ has no internal joints, actuator limits are also replaced with approximating constraints. For example, joint velocity limits can be approximated by Cartesian velocity limits on body-fixed contacts (e.g. feet or hands) via

$$||\mathbf{v}_i(t)|| \leq \mathbf{v}_{i,\max}. \tag{6}$$

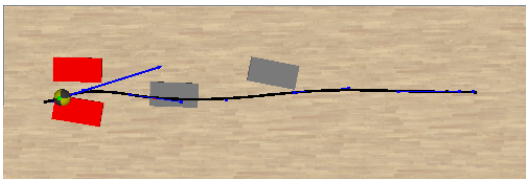
IV. Trajectory Primitives

We use low-dimensional trajectory primitives to solve the simplified planning problem. A trajectory primitive is a short, time-indexed, feasible trajectory for the robot $\hat{\mathcal{R}}$ – that is, a rigid-body position and orientation trajectory and a sequence of active contacts that satisfies the contact constraints. The optimal time scale and boundary characteristics for these primitives are not considered here; for our examples, suitable primitives were determined subjectively.

As a trajectory for $\hat{\mathcal{R}}$, each primitive m is annotated with its a nominal pose and velocity at its start ($m.\mathbf{r}_s$, $m.\mathbf{v}_s$) and the end ($m.\mathbf{r}_e$, $m.\mathbf{v}_e$). Available primitives are assembled into a library $m \in M$ for use in planning (see Section V).



(a) A "long jump" primitive designed from a YouTube video.



(b) A "start to run" primitive taken from motion capture data.

Fig. 3. Examples of trajectory primitives taken from several sources.

A. Sources of Trajectory Primitives

Due to their low dimensionality, trajectory primitives are intended to be easily learnable from a diverse array of sources. For example, they can be inferred from video clips, sketched through a computer interface, taken from motion capture data, or even described verbally by an expert. Recent advances in inexpensive joint-tracking depth sensors developed for video game consoles may allow an autonomous robot to learn complex trajectory primitives from human demonstrators.

In this paper, we used primitives of starts, stops, runs, and jumps. These primitives were either taken from motion capture data[2] or manually designed from human running data and video clips.[3] For examples of primitives used here, see Figure 3.

B. Trajectory Primitive Representation

Each primitive is split into segments around changes in contact state. Therefore, each segment of a primitive has a fixed active contact configuration.

We choose to represent the center-of-mass trajectory for each segment as cubic Bézier curve. This representation has a number of advantages: (a) smooth curves can be represented compactly with nodes and handles, (b) flight phases with constant vertical acceleration can be represented simply with quadratic segments, (c) nodes present easy ways to split and compose primitives, and (b) designing trajectories by humans is easy due to their popularity in popular drawing software.

While the center-of-mass trajectory has continuous position and velocity across segment boundaries, this representation does produce discontinuous jumps in acceleration. This is reasonable because a new contact configuration leads to a corresponding discontinuity in the constraints on the contact forces.

[2]Motion capture data provided by the *CMU Graphics Lab Motion Capture Database*, (`http://mocap.cs.cmu.edu/`).

[3]A video clip of a long jump taken from YouTube video *amazing parkour jump!* (`http://www.youtube.com/watch?v=LsJYxJuwU2w`).

V. Sampled Composition A* Planner

We are now tasked with planning for the simplified robot $\hat{\mathcal{R}}$ through an environment using our library of trajectory primitives. Because the force of gravity is uniform across all primitives, they are invariant to 4 dimensions of position and orientation (i.e. x, y, z, and yaw). This allows a small library of primitives to be composed to form longer paths. However, each primitive has fixed, continuous-valued contact locations and start and end states. Therefore, exact composition of our finite set of primitives will generally be inadequate for solving a locomotion problem.

One approach is to parameterize each primitive to allow a volume of feasible start and/or end states and contact locations around the nominal points. However, this approach has disadvantages, including the increased complexity of the primitives and the corresponding difficulty to design and/or learn them. Therefore, we take a different approach. We extend the heuristic-based A* graph search algorithm to allow sampled composition of actions.

A. Description of Sampled Composition A*

The Sampled Composition A* algorithm (SCA*) is a variant of the traditional A* heuristic-based graph search planner applied to a continuous state space with a discrete set of actions. The state represented at each node is augmented with an additional real-valued *sampling variance* σ^2. When invoking the planner from a start state $\mathbf{s}_s$ to a goal state $\mathbf{s}_g$, the open set is initialized with a single node n_0 at the start state with a zero variance:

$$n_0.s \leftarrow s_s;\ n_0.\sigma^2 \leftarrow 0. \tag{7}$$

SCA* proceeds similarly to A*, with the addition of a state sampling function $\text{SCASAMP}(\mathbf{s}, \sigma^2)$, functions that define the variance schedule $\text{SCAVARINC}(\sigma^2)$ and $\text{SCAVARDEC}(\sigma^2)$, and a variance-increase edge cost function $\text{SCAVAREDGECOST}(\sigma^2)$. The node expansion function (described in Function 1) is expanded to make use of these new functions to sample new states and manage each node's augmented variance.

During each expansion, an additional child node n_v is added with the same state as the parent node, with an increased variance (line 1). An adjusted state $\mathbf{s}_a$ is sampled from a distribution around the parent node's state given its variance (line 3). Each action is then considered when appended to this new adjusted state (in the EXPANDNODE() function, line 4). The variances of the child nodes at the new post-action states are deceased. For a simple 1D example, see Figure 4.

Edge costs are unaffected, with the exception of each edge to an increased-variance node (as created in 1). These edges are given a cost in relation to the variance increase across the edge.

B. Search with Trajectory Primitives using SCA*

By using SCA*, we allow our planner to make inexact compositions of trajectory primitives by sampling nearby states. During planning, this manifests itself as discontinuous jumps in position and velocity at composition points. These jumps incur cost in relation to their size, and our heuristic-based graph search planner reasons about these costs during planning. Once a plan is found, primitive endpoints are manually adjusted to match, and an optimizer is used to pursue feasibility (see Section VI).

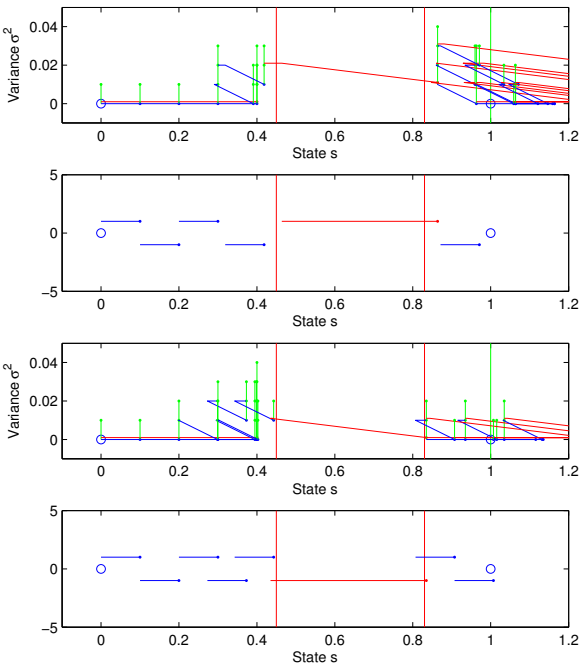


Fig. 4. Two solutions to a simple 1D "run-and-jump" example problem solved with Sampled Composition A*. There are two primitives: a run (blue) and a jump (red). A gap is marked in red. The first two plots represent one trial, and the second two show the result from a second run.

We use simple definitions for SCA*:

- s: simplified robot $\hat{\mathcal{R}}$ pose and velocity $\hat{\mathbf{x}}$, along with the most recent primitive used m
- $\text{SCASAMP}(s, \sigma^2)$: pose sampled from a normal distribution with variance σ^2
- $\text{SCAVARINC}(\sigma^2) = \sigma^2 + \delta_{\sigma^2}$
- $\text{SCAVARDEC}(\sigma^2) = \max(\sigma^2 - \delta_{\sigma^2}, 0)$
- $\text{SCAVAREDGECOST}(\sigma^2) = c_\delta$

The function we use to compose primitives from a given state is described in Function 2. From the parent state s_a, we consider all possible primitives M. We compose each

Function 1 $N_c \leftarrow \text{SCASTAREXPANDNODE}(n_p)$

1: $n_v.s \leftarrow n_p.s;\ n_v.\sigma^2 \leftarrow \text{VARINC}(n_p.\sigma^2)$
2: $N_c \leftarrow \{n_v\}$
3: $s_a \leftarrow \text{SAMP}(n_p.s, n_p.\sigma^2)$
4: $S_c \leftarrow \text{EXPANDNODE}(s_a, n_p.\sigma^2)$
5: **for all** s **in** S_c **do**
6: $\quad n_i.s \leftarrow s;\ n_i.\sigma^2 \leftarrow \text{VARDEC}(n_p.\sigma^2)$
7: $\quad N_c \leftarrow N_c \cup \{n_i\}$
8: **end for**
9: **return** N_c

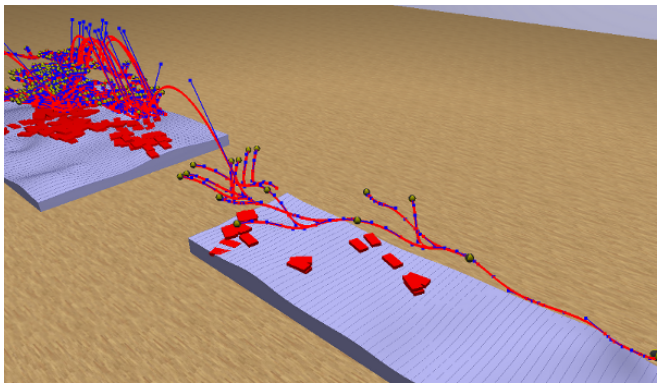


Fig. 5. SCA* search running on an example jump problem. Multiple expanded nodes are shown at different sampled poses.

primitive m onto the adjusted parent rigid-body state $s_a.\hat{\mathbf{x}}$. If the resulting state discontinuity is beyond a variance-dependent threshold with respect to metric M_x, the primitive is not considered. Next, each requisite footstep is snapped to its nearest environment location; if the requisite difference is beyond a similar threshold with respect to M_f, the primitive is not considered. If the composition is deemed acceptable, it is added to the list of child states to be returned. For a visual representation of SCA* search running, see Figure 5.

Function 2 $S_c \leftarrow \text{EXPANDNODE}(s_a, \sigma^2)$

$S_c \leftarrow \{\}$
for all m **in** M **do**
  $m_c = \text{COMPOSE}(m, s_a)$
  if $||m_c.\hat{\mathbf{x}}_s - s_a.\hat{\mathbf{x}}||_{M_x} > \sigma^2$ **then**
    continue // skip this primitive
  end if
  if $||f.\mathbf{r} - \text{ENV}(f.\mathbf{r})||_{M_f} > \sigma^2$ **for any** $f \in m_c.F$ **then**
    continue // skip this primitive
  end if
  $s_i.\hat{\mathbf{x}} \leftarrow m_c.\hat{\mathbf{x}}_e$
  $s_i.m \leftarrow m_c$
  $S_c \leftarrow S_c \cup \{s_i\}$
end for
return S_c

C. Enforcing Continuity of Composed Primitives

The SCA* planner finds a candidate sequence of trajectory primitives from the start node to the goal. However, due to the sampling nature of the primitive composition, there will be discontinuities in the rigid-body trajectory.

The sequence of primitives found by SCA* is made continuous by averaging the positions and velocities at each discontinuity (see Figure 6). This eliminates discontinuities in the trajectory. If newly-adjacent segments share an identical contact configuration (e.g. flight phase), these segments are also merged together.

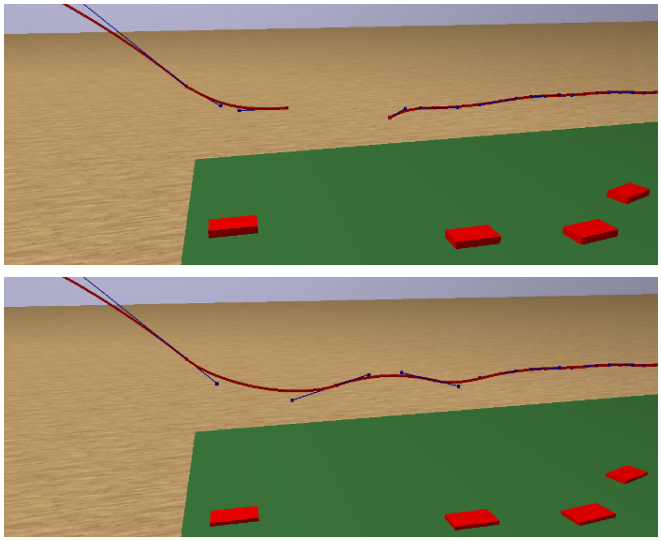


Fig. 6. Example of the joining procedure to produce a continuous plan from a sequence of trajectory primitives.

VI. Trajectory Optimizer

While individual trajectory primitives may obey the necessary external constraints (2), the sampling nature of SCA* may cause segments of the composed plan to be infeasible. We use a simple trajectory optimizer to ensure feasibility. While success is not guaranteed, separately optimizing several candidate solutions as described in Section I provides a high likelihood of success.

The time-indexed trajectory ξ consists of states $\hat{\mathbf{x}}$ comprising the motion of the rigid body and the locations and timings of environment contacts. In our case, ξ is parameterized like a trajectory primitive, as described in Section IV.

A. Instantaneous Cost Function

The instantaneous cost along the trajectory ξ at time t is the sum of several terms: the contact force magnitude cost, the contact force constraint cost, and the approximation costs for internal constraints and reachability.

$$c(\xi, t) = c_f(\xi, t) + c_e(\xi, t) + c_a(\xi, t) \tag{8}$$

The cost $c_f(\xi, t)$ penalizes large contact forces. Each contact-specific weighting matrix $\mathbf{W}_{fi}$ may be time-varying. Each contact force $\mathbf{f}_i$ is also time-varying, and is determined below in Section VI-B.

$$c_f(\xi, t) = \frac{1}{2} \sum_{i=1}^{n_c} \mathbf{f}_i^T \mathbf{W}_{fi} \mathbf{f}_i \tag{9}$$

The cost $c_e(\xi, t)$ penalizes insufficient foot placement. It acts if a given force distribution $\mathbf{f}_i$ is unable to produce the required contact force $\mathbf{f}_c = \dot{\mathbf{h}} - \mathbf{f}_g$, from (3). Note the discontinuity introduced by the constant c_{e0} which biases the optimizer to exactly satisfy the constraint.

$$c_e(\xi, t) = \begin{cases} 0 & \text{if } \Sigma_i \mathbf{f}_i = \mathbf{f}_c \\ c_{e0} + \frac{1}{2}(\Sigma_i \mathbf{f}_i - \mathbf{f}_c)^{\mathrm{T}} \mathbf{W}_e (\Sigma_i \mathbf{f}_i - \mathbf{f}_c) & \text{otherwise} \end{cases} \tag{10}$$

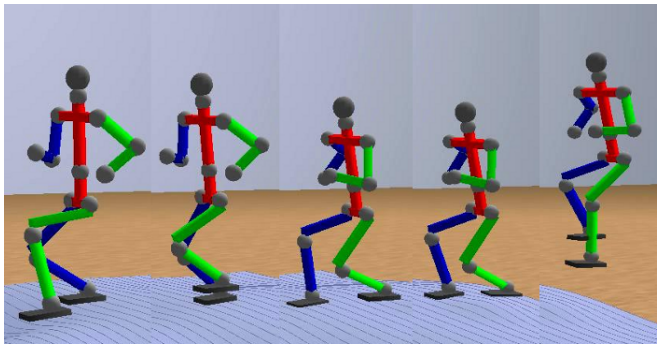


Fig. 7. Frames taken from a jump solution created by our planner. Here, the figure is traveling from right to left. It lands from a jump, and takes two steps while traveling at approximately 4 m/s.

The approximation cost $c_a(\xi, t)$ penalizes violations of the corresponding constraints (e.g. 5).

B. Instantaneous Contact Force Distribution

Note that the cost components c_f and c_e are also functions of the particular force distribution $\mathbf{f}_i$ over the available contacts. We use Quadratic Programming to minimize the cost subject to the external constraints (2).

$$c(\xi, t) = \min_{\mathbf{f}_i} \left[c_f(\xi, t, \mathbf{f}_i) + c_e(\xi, t, \mathbf{f}_i) \right] + c_a(\xi, t) \quad (11)$$

C. Optimization

The parameterized trajectory ξ is then optimized with respect to the full trajectory cost,

$$c(\xi) = \int_t c(\xi, t) dt. \quad (12)$$

The optimizer is allowed to alter the full rigid-body trajectory as well as the footstep locations and timings. We have found that simple local search is sufficient to find a feasible trajectory within a few minutes on modern hardware.

VII. Mapping to a Full-Body Plan

Once we find a feasible trajectory ξ for the simplified robot $\hat{\mathcal{R}}$ from $\hat{\mathbf{x}}_s$ to $\hat{\mathbf{x}}_g$, we need to map this to a full-dimensional trajectory from $\mathbf{x}_s$ to $\mathbf{x}_g$. Due to the nature of the simplification described in Section III, the difficult part of the solution (finding a center-of-mass motion and feasible contact locations) is already solved.

What remains is primarily a kinematic problem: how to actuate the robot's internal degrees of freedom to achieve and maintain the body-fixed contacts c_i in the necessary locations, while simultaneously maintaining the correct full-body orientation.

A. Regulating Full-Body Orientation

Recall from Section III that while momentum equivalence guarantees that the linear position and velocity of the centers of mass of $\mathcal{R}$ and $\hat{\mathcal{R}}$ will coincide, it makes no such assurances of the orientations or angular velocities of bodies. Here, we would like to actuate internal joints in order to maintain a selected body b (e.g. a pelvis or torso) at the same orientation as the simplified robot $\hat{\mathcal{R}}$.

The total system momentum $\mathbf{h}$ of $\mathcal{R}$ is the sum of the momenta of the component bodies,

$$\mathbf{h} = \sum_i \mathbf{I}_i \mathbf{v}_i, \quad (13)$$

with $\mathbf{I}_i$ the body's spatial inertia matrix, and $\mathbf{v}_i$ its spatial velocity.

We write each body's velocity $\mathbf{v}_i$ relative to the selected body's velocity $\mathbf{v}_b$ using a Jacobian and the internal joint velocities $\dot{\mathbf{q}}_{int}$,

$$\mathbf{h} = \sum_i \mathbf{I}_i \left(\mathbf{v}_b + {}^b\mathbf{J}_i \, \dot{\mathbf{q}}_{int} \right), \quad (14)$$

$$\sum_i \mathbf{I}_i \, {}^b\mathbf{J}_i \, \dot{\mathbf{q}}_{int} = \mathbf{h} - \sum_i \mathbf{I}_i \, \mathbf{v}_b. \quad (15)$$

Therefore, given an instantaneous system momentum $\mathbf{h}$ and a desired selected body velocity $\mathbf{v}_b$ (both given by the simplified trajectory ξ), (15) imposes the linear orientation-regulating constraint.

B. Imposing Body-Fixed Contact Trajectories

The simplified plan ξ gives the locations and timings of contact, but does not specify swing trajectories. We use a simple interpolation function to generate these trajectories. Each contact imposes an additional constraint on the space of permissible internal joint trajectories,

$${}^b\mathbf{J}_{ci} \, \dot{\mathbf{q}}_{int} = \mathbf{v}_{ci} - \mathbf{v}_b. \quad (16)$$

C. Solving using Quadratic Programming

Along with the equality constraints (15) and (16), we can also enforce joint/actuator position and velocity constraints with linear inequalities on $\dot{\mathbf{q}}_{int}$. While solving for a full-body trajectory for $\mathcal{R}$, we can also apply a quadratic cost to $\Delta\dot{\mathbf{q}}_{int} = \dot{\mathbf{q}}_{int} - \dot{\mathbf{q}}_{int,last}$ to effectively minimize joint accelerations.

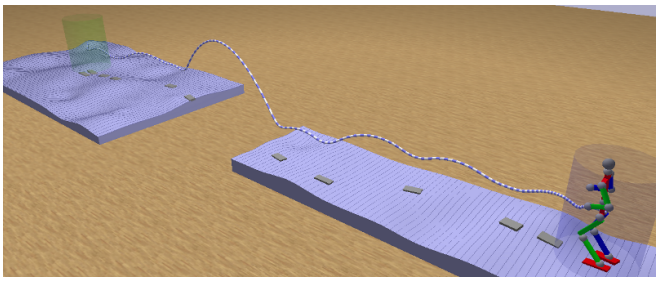


Fig. 8. An example solution of a jump produced by our planner. For an overview of the planning and optimization process, see the attached video.

VIII. Discussion and Future Work

This paper presents a framework and algorithm for solving extreme locomotion planning problems using trajectory primitives. The full-dimensional system is reduced by momentum equivalence to a simplified system. A heuristic-based graph search planner (e.g. Sampled Composition A*) is used to compose learned trajectory primitives for the reduced system into a candidate path. The resulting trajectory is then optimized to pursue feasibility, and subsequently mapped back to the full-dimensional system.

Throughout the planning hierarchy, the framework does not guarantee feasibility, either as primitives are composed or as the resulting plan is mapped to the full-dimensional system. While explicitly parameterizing primitives as discussed in Section V may mitigate this problem, we have found that simultaneously generating, optimizing, and mapping multiple candidate solutions provides good results for the problems we have considered. While this may appear costly, it is inherently parallel, and therefore a good match for modern computer hardware. A future extension to this work, in common with other hierarchical planners, would be to generate a set of sufficiently different plans and/or to allow errors to be propagated to earlier planning stages.

We created an implementation of the SCA* planner, trajectory optimizer, and mapper to a full-body dynamic model. We tested the framework on several example jump problems; solutions consisted of five to ten composed and adjusted primitives. Running times for the planner and optimizer are approximately 5 minutes each on modern hardware; we anticipate that substantial speed improvements are possible through parameter tuning, optimizer choice, and parallelization.

Exploitation of sparse-searching or anytime variants of A* (e.g. [23]) would improve the performance of the planner. The planner's domain can be extended by also considering task forces, time-varying environments, an energy-loss impact model, and rolling/sliding contact. Algorithms for obstacle avoidance (e.g. [24]) for generating body and swing trajectories would also extend the domain.

The optimizer can also be extended to consider plan robustness. While the plans generated by this framework are physically feasible in simulation, more work is required in the areas of state estimation and robustness in order to approach a physical implementation.

The choice of planning horizon length is an important aspect of this approach that should be considered in the future. The framework as presented here considers the entire problem, and plans for the full trajectory before execution. The problems we've considered so far have solutions lasting less than 10 seconds and with tens of footsteps. For longer problems, this approach will become quickly intractable. In contact, a purely reactive approach will necessarily suffer poor performance, and in many cases be unable to find solutions. Finding a suitable method for receeding horizon planning is essential for solving a broader class of problems.

References

[1] B. van Hoytema, "Parkour voor beginners," Image from Flickr, Creative Commons BY-NC-SA 2.0 License.

[2] M. Vukobratović and D. Juričić, "Contribution to the Synthesis of Biped Gait," IEEE Transactions on Biometical Engineering, Jan 1969.

[3] M. Vukobratović, and B. Borovac, "Zero-Moment Point – Thirty Five Years of its Life," International Journal of Humanoid Robotics (2004), 1(1), pp 157173.

[4] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep Planning for the Honda ASIMO Humanoid," *2005 IEEE International Conference on Robotics and Automation*, April 2005.

[5] J. Buchli, J. Pratt, and N. Roy, "Special issue on Legged Locomotion," *International Journal of Robotics Research*, (30)2, Feb 2011.

[6] J. Chestnutt, "Navigation Planning for Legged Robots," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA 15213.

[7] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, Robust Quadruped Locomotion over Challenging Terrain," *2010 IEEE International Conference on Robotics and Automation*, pp 2665-2670, May 2010.

[8] I. Mordatch, M. de Lasa, and A. Hertzmann, "Robust Physics-Based Locomotion using Low-Dimensional Planning," *ACM Transactions on Graphics*, 29(4), 2010.

[9] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-Stable Motion Planning for Humanoid Robots," *Autonomous Robots*, 12(1), pp 105-118, Jan 2002.

[10] S. Takao, Y. Yokokohji, and T. Yoshikawa, "FSW (Feasible Solution of Wrench) for Multi-Legged Robots," *2003 IEEE International Conference on Robotics and Automation*, 3, pp 3815-3820, Sep 2003.

[11] T. Bretl, "Multi-step motion planning: Application to free-climbing robots," Ph.D. dissertation, Stanford University, Stanford, CA, 2005.

[12] K. Hauser, "Motion Planning for Legged and Humanoid Robots," Ph.D. dissertation, Stanford University, Stanford, CA, 2008.

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

[14] C. Collette, A. Micaelli, C. Andriot, and P. Lemerle, "Dynamic Balance Control of Humanoids for Multiple Grasps and non Coplanar Frictional Contacts," *2007 IEEE-RAS International Conference on Humanoid Robots*, pp 81-88, 29 Nov 2007 - 1 Dec 2007.

[15] C. Collette, A. Micaelli, C. Andriot, and P. Lemerle, "Robust Balance Optimization Control of Humanoid Robots with Multiple non Coplanar Grasps and Frictional Contacts," *2008 IEEE International Conference on Robotics and Automation*, pp 3187-3193, May 2008.

[16] J. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard, "Interactive Control of Avatars Animated with Human Motion Data," *ACM Transactions on Graphics*, 21(3), July 2002.

[17] M. Choi, J. Lee, and S. Shin, "Planning Biped Locomotion using Motion Capture Data and Probabilistic Roadmaps," *ACM Transactions on Graphics*, 22(2), pp 182-203, 2003.

[18] P. Reitsma and N. Pollard, "Evaluating Motion Graphs for Character Animation," *ACM Transactions on Graphics*, 26(4), art 18, Oct 2007.

[19] A. Safonova and J. Hodgins, "Construction and Optimal Search of Interpolated Motion Graphs," *ACM Transactions on Graphics*, 26(3), July 2007.

[20] L. Liu, K. Yin, M. van de Panne, T. Shao, and W. Xu, "Sampling-based Contact-rich Motion Control," *ACM Transctions on Graphics*, 29(4), 2010.

[21] A. Degani, "Minimalistic Dynamic Climbing," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, 2010.

[22] R. Altendorfer, N. Moore, H. Komsuoglu, M. Buehler, H. B. Brown Jr., D. McMordie, U. Saranli, R. Full, and D. E. Koditschek, "RHex: A Biologically Inspired Hexapod Runner," *2001 Autonomous Robots*, 11(3), pp 207-213, Nov 2001.

[23] M. Likhachev and A. Stentz, "R* Search," *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2008.

[24] N. Ratliff, M. Zucker, J. Bagnell, and S. Srinivasa, "CHOMP: Gradient Optimization Techniques for Efficient Motion Planning," *2009 IEEE International Conference on Robotics and Automation*, pp 489-494, May 2009.