Towards Computationally Efficient Planning of Dynamic Multi-Contact Locomotion

Gray C. Thomas and Luis Sentis

***Abstract*— This paper considers the problem of numerically efficient planning for legged robot locomotion, aiming towards reactive multi-contact planning as a reliability feature. We propose to decompose the problem into two parts: an extremely low dimensional kinematic search, which only adjusts a geometric path through space; and a dynamic optimization, which we focus on in this paper. This dynamic optimization also includes the selection of foot steps and hand-holds—in the special case of instantaneous foot re-location. This case is interesting because 1) it is a limiting behavior for algorithms with a foot switching cost, 2) it may have merit as a heuristic to guide search, and 3) it could act as a building block towards algorithms which do consider foot transition cost. The algorithm bears similarity both to phase space locomotion planning techniques for bipedal walking and the minimum time trajectory scaling problem for robot arms. A fundamental aspect of the algorithm's efficiency is its use of linear programming with reuse of the active set of inequality constraints. Simulation results in a simplified setting are used to demonstrate the planning of agile locomotion behaviors.**

I. Introduction

Dynamic multi-contact locomotion is easy to take for granted as a human, but the simple ability to opportunistically use hands in addition to feet to balance has been a challenge for humanoid robots. Robots which can safely enter the home must be practical and robust—robots which can plan for dynamic multi-contact locomotion fast enough to recover from disturbances like slipping, or being kicked. Such planning is not possible today, and highlights the importance of planning efficiently. To this end we investigate an area between kinodynamic planning, which is general but computationally complex, and flat-ground locomotion planning, which is simple and fast but not applicable to the general multi-contact case.

We aim to bridge this gap with a nested optimization strategy—a *kinematic optimization* and a *dynamic optimization*. This paper only addresses the dynamic optimization, but the general strategy for both is explained below. By kinematic optimization we mean a process which chooses kinematic paths of the center of mass of the robot. These paths must be continuous, so Bezier splines are a natural parameterization.

Since any path only represents the center of mass of the robot, the path still contains uncertainty—a family of paths in joint space will all achieve the same center of mass path.

G. Thomas and L. Sentis are with the Department of Mechanical Engineering, University of Texas at Austin, Austin, TX, 78712 USA. corresponding e-mail: gray.c.thomas@gmail.com.

This work was partially supported by NASA Space Technology Fellowship #NNX15AQ33H.

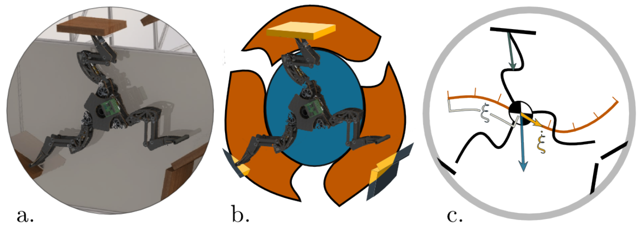

Fig. 1. **Example robot, conservative collision model, and path-following multi-contact point mass model.** a.) Rendering of a simplistic multi-contact robot in a 2D environment. Our model is the least over-conservative with robots that have 1) a heavy body and light legs, 2) wide leg workspaces, 3) fast legs, 4) approximately uniform leg-tip force limits over that workspace, and 5) legs that can retract towards the body. The pictured "triskelion" robot was designed to demonstrate aggressively dynamic multi-contact, and is a good fit for this model—but the model is also applicable to humanoids. b.) Work spaces of the robot. Workspaces are used to identify potential collisions and potential contacts. The terrain which falls in the potential contact region has been highlighted. If any terrain intersected the blue region then there would be potential for a collision even if the robot retracted its legs. c.) Diagram of path-following multi-contact point mass model with one leg in contact and two legs not in contact. Here the robot attempts to follow a center of mass path (orange), and its position is represented as progress along this path in the arc-length variable ξ. The speed of path following, $\dot{\xi}$, is always positive. The model allows the robot to maximize path acceleration, $\ddot{\xi}$, without deviating from the path—by choosing where the feet are, and what the reaction forces are at each foot. It is impossible to move the legs instantaneously, so a non-zero swing time must be accounted for later.

Thus when a kinematic optimization checks for collisions it uses a center of mass based model which represents this ambiguous behavior. In spirit, this is a trick from the field of robust control: Use a simple model instead of a complex one, but embed uncertainty into the simple model to account for the simplification.

A kinematic optimization relies on a dynamic optimization to evaluate the dynamic quality of a kinematic path. Using this information such an optimization searches for the dynamically best path which avoids potential collisions.

This dynamic optimization problem is similar to a reactive planning algorithm in the assumptions it makes about the robot. It chooses the speed at which the robot follows the path, and also where each leg should go. The trade-off sacrificed is real optimality in the resulting motions, and this stems from conservative assumptions for the kinematic and dynamic models.

Assuming that the problem of optimizing paths given an arbitrary quality criterion has already been addressed, we present an algorithm for efficient computation of a limiting case of the dynamic optimization, the dynamics and leg-allocation problem. We leave out the problem of

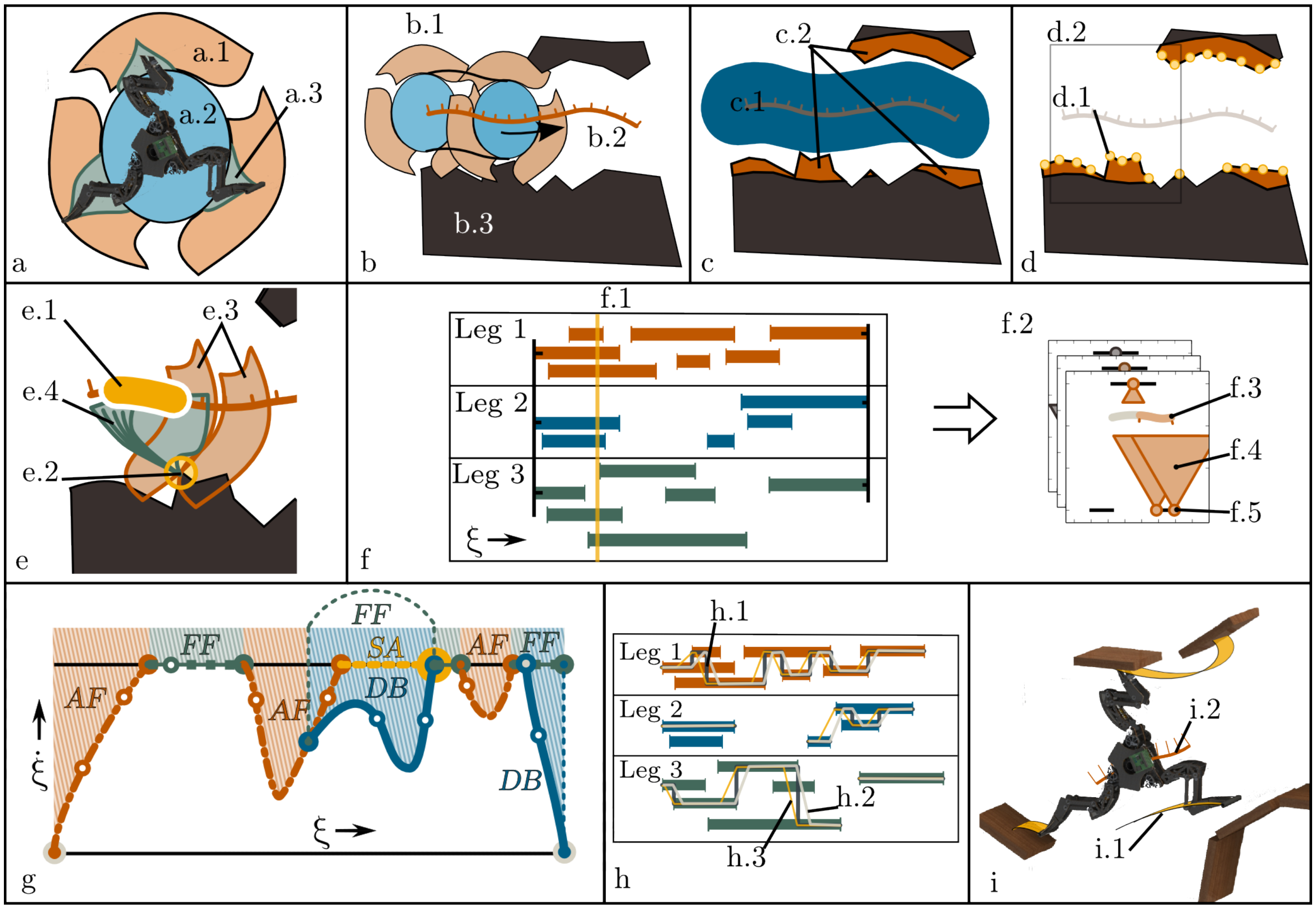

Fig. 2. **Big picture planning framework.** a) Uncertain robot collision model with a.1) workspaces of the limbs, a.2) worst-case body collision model and a.3) safe foot retraction workspace. b) Sweeping b.1) the robot-centric collision model along b.2) the center of mass path to check for intersections with b.3) the terrain. c) Result of sweeping procedure when the path is good with c.1) no terrain in the potential collision region and c.2) terrain identified as reachable. d) Contacts only considered within reachable region. d.2) The area around d.1) a specific contact is the focus of e) the process of calculating the contact limit. The e.1) ξ region in which e.2) a contact is usable (valid) is defined by e.3) the leg workspace and e.4) the the safe leg retraction workspace. At any ξ the contact can be invalidated by not being in the leg workspace, and also by terrain obstructing the safe retraction workspace. f) Contact options for various feet represented by their regions of validity. Contacts are combined into stances combinatorially. For example f.1) marks a point along the path with 3 options for leg 1, 2 options for leg 2 and 2 options for leg three, adding in the option to have each leg lifted there are 24 separate stance options available at this time. Contacts are combined combinatorially into f.2) a big list, where each stance has f.3 a region over which it is valid, and f.4) a friction cone approximation for each f.5) contact. g) Minimum time planning can now be conducted as in Sec. IV. h) After the planning, or perhaps as a modification to it, transition times for each leg must be forced. h.1) The limiting case of instantaneous transitions could be converted to a valid path through leg-allocation space by h.2) naïvely switching after or h.3) just before the nominal transition time—the instant that one stance becomes better than another. Finally i) application on a robot will require generating i.1) swing trajectories which, along with i.2) the center of mass trajectory will be tasks for a whole body controller to follow. Note that this only specifies the locomotion portion of the robot's behavior, and that additional active sensing or manipulation tasks are decoupled from the locomotion planning.

designing a good heuristic to ensure adequate transition time between stances. This means the algorithm presented here does not, by itself, generate plausible motions, but rather a theoretical limiting case—the limit as leg transitions become instantaneous. However, this is not to say that the algorithm presented here is without use. It could certainly prove to be an important building block towards efficient algorithms possessing such a heuristic. It is additionally capable of invalidating paths, since the instantaneous foot motion assumption is a relaxation of the real problem—and this suggests potential for use as a planning heuristic.

Additionally, the implementation presented here is at an early stage of development, and must eventually validate the overarching planning structure described in Fig. 2. Algorithm aside, this paper 1) introduces the path-following multi-contact point mass as a model of dynamic multi-contact locomotion, 2) presents a generalization of the minimum time algorithm as a single hybrid dynamical system—compatible with efficient hybrid dynamic system integration software [1], and 3) uses the phase space framework to explain interesting behaviors which will naturally exist in all dynamic multi-contact locomotion problems.

II. Related Work

The motion planning community has addressed multi-contact locomotion in a general manner. This community relies on powerful algorithms which can solve virtually any planning problem given enough time, [2], [3]—probabilistic road-maps (PRM) [4] and rapidly exploring random trees (RRT) [5]. It could be said that the approach of PRM and RRT is to make intractably hard problems tractable by sacrificing the optimality of the path, substituting it with the best path in an approximate graph (PRM) or simply the first path found (RRT). Multi-contact planning has been accomplished for simple static climbing systems using off-line computation of a probabilistic road map [6]. Off-line computation of dynamic trajectories for complex robots is very difficult, but has been accomplished for humanoids [7], and leg-wheeled space robots [8]. In such cases the best results have exploited good guesses to predispose the planner to try the best paths first, *e.g.* a precomputed set of walking motion primitives [7]. Despite this success, these approaches remain practical only for off-line optimization. Planning is fundamentally more difficult in high dimensional spaces, and both PRM and RRT suffer from this curse of dimensionality.

Another branch of motion planning—which has seen renewed interest recently—plans dynamics once a kinematic motion has already been specified [9]–[11]. While originally envisioned as a method to improve industrial efficiency for robot arms, this approach can be applied to the general problem of optimizing kinodynamic trajectories as well [12]. For this purpose it acts as a *dynamic optimization* similar to our vision, but without the flexibility to choose footsteps. Naturally, this means the kinematic trajectory fully specifies the robot position. A minimum speed traversal algorithm has recently been applied to multi-contact humanoid maneuver planning [13] given a full joint space motion path. There have also been attempts to extend it to more general classes of motion planning: combining it into an RRT framework [14]; and using a convex optimizer to offer cost functions other than minimum time [15]. The latter again applied the framework to multi-contact humanoid maneuver planning in the full joint space of the robot. As shown in [13], however, the minimum time algorithm is much faster than the convex optimization approach for the subset of cases which can be solved by both.

The fastest planning occurs with the simplest models, and in that category the flat-ground locomotion models are clear winners. These models abstract themselves above the joint space of the robot. They deal only with the center of mass and the location of the feet. The ZMP oriented cart-table model [16], for instance, specifies a relationship between the feet and the acceleration of the center of mass, given the simplifying assumption that the center of mass maintains a constant height. The linear inverted pendulum model makes the same assumption to explain the unstable dynamics of balancing on a point foot [17] and presents analytically solvable center of mass dynamics. This is simplified even further by the capture point [18], which is extended to 3D by the divergent component of motion [19] locomotion models. Both of which provide analytical frameworks relating potential footstep locations to the center of mass behavior. All of these models abstract themselves above robot kinematics to yield simple and easy-to-compute results.

One less analytic, yet still very efficient, model of locomotion is the prismatic inverted pendulum model [20]. This 2D model assumes that the center of mass will follow a pre-determined path, and it reasons about when to switch between point-foot contacts using the phase space of the center of mass—the plot of its velocity against its position. This model's main strength is that it allows the path to be determined beforehand, rather than arising as a result of the footstep locations and linear-in-position leg forces—as is the case with the capture point, divergent component of motion, and linear inverted pendulum model. An application of phase space planning to robot control used this model's prediction of the future to reactively determine footstep locations [21]–[23]. This illustrates the real-time potential of these phase space techniques. It has also been extended to a more ambitious numerical optimization process—one which both adjusts the center of mass path and searches for point-footstep locations along a 1D ground surface—to maximize efficient motion over rough terrain [24]. Which demonstrates it acting as a *dynamic optimization* subcomponent of a larger *kinematic optimization*.

III. A Path-Following Multi-Contact Point Mass

We now introduce some notation describing our simple model for analyzing multi-contact locomotion constrained to follow a center of mass path. This path is parameterized by a progression variable[1] $\xi \in \Xi = [0, \xi_{\max}]$ (Ξ is a closed set of reals), with time derivative denoted $\dot{\xi} \in \mathbb{R}_+$ (real non-negative numbers). The multi-contact stances are represented by a set $\mathfrak{S}$, where each stance $\mathcal{S} \in \mathfrak{S}$ is itself a set of indexes. Not all stances are valid, in the sense that the kinematic model claims them to be certainly reachable and not at risk of tripping, at any location along the path. We define the boolean function $\texttt{valid} : \mathfrak{S} \times [0, \xi_{\max}] \mapsto \{\mathsf{T}, \mathsf{F}\}$ such that the set of all stances which are valid at a location ξ is $\mathfrak{V}_\xi = \{S : \text{valid}(\mathcal{S}, \xi) = \mathsf{T}\}$.

We define the robot model—a (non-rotating) point mass, constrained to follow a path, and limited to a maximum speed—using mass $m \in \mathbb{R}_+$, (kinematic) path $\mathbf{x} : \Xi \mapsto \mathbb{R}^3$, gravitational acceleration vector $\mathbf{g} \in \mathbb{R}^3$, and maximum speed $s_{\max}$. The path is differentiable twice both by ξ ($\mathbf{x}'(\xi) \in \mathbb{R}^3, \mathbf{x}''(\xi) \in \mathbb{R}^3$) and by time ($\dot{\mathbf{x}} \in \mathbb{R}^3, \ddot{\mathbf{x}} \in \mathbb{R}^3$). We choose to parameterize this path such that ξ is the arclength of the path and $\|\mathbf{x}'\| = 1$. A net force $\mathbf{f}$ acts on the center of mass.

Each contact in each stance in the set of stances, index $i \in \mathcal{S} \in \mathfrak{S}$, also has a position $\mathbf{x}_i \in \mathbb{R}^3$ and a force $\mathbf{f}_i \in \mathbb{R}^3$. Each contact's reaction force must remain inside a friction cone. We approximate the friction cone as the convex cone generated by a finite number $n_i \in \mathbb{N}$ of basis vectors $\mathbf{b}_{j,i} \in \mathbb{R}^3 \quad j = 1, ..., n_i$. We combine these into $\mathbf{B}_i \in \mathbb{R}^{3 \times n_i}$, a matrix with jth column $\mathbf{b}_{j,i}$. The generation of this cone requires positive multiplier variables $\phi_{j,i} \in \mathbb{R}_+ \quad j = 1, ..., n_i$. We consider $\boldsymbol{\phi}_i \in \mathbb{R}_+^{n_i}$ a column vector, and $\boldsymbol{\Phi}_\mathcal{S}$ the vertical stacking of all such

[1]Analogous to s in the work of [9].

column vectors in stance $\mathcal{S}$. Ultimately, the force for each contact $\mathbf{f}_i = \sum_{j=1}^{n_i} \mathbf{b}_{j,i}\phi_{j,i} = \mathbf{B}_i\boldsymbol{\phi}_i$. The multipliers are our optimization variables, and are constrained by

$$\phi_{i,j} \geq 0 \ \forall \ i \in \mathcal{S}, \ j \in \{1, ..., n_i\} \tag{1}$$

$$\sum_{j\in\{1,...,n_i\}} \phi_{i,j} \leq 1 \ \forall \ i \in \mathcal{S}. \tag{2}$$

Thus, (1) ensures that the forces are within the friction cones, and (2) ensures that contact forces do not exceed a pre-defined limit.

As is shown in [25], the relationship between the center of mass acceleration, whole body angular momentum rate of change, and net reaction wrench can be separated from the dynamics of the robot's joints and the internal wrenches. Though whole body motion occasionally includes significant rotation, we follow other simple locomotion centric models [16]–[18] which treat angular momentum as non-essential. This naturally leads to a point mass description of the robot, and a constraint on the reaction forces which guarantees there is no net reaction moment about the center of mass. The net reaction wrench $\mathbf{f}$ will thus determine the acceleration of the center of mass point $\mathbf{x}$,

$$m\ddot{\mathbf{x}} = \mathbf{f} = \sum_{i\in\mathcal{S}} \mathbf{B}_i\boldsymbol{\phi}_i + m\mathbf{g}, \tag{3}$$

$$0 = \sum_{i\in\mathcal{S}} [\mathbf{x}_i - \mathbf{x}]_\times \mathbf{B}_i\boldsymbol{\phi}_i \tag{4}$$

where m is the robot mass and $[\cdot]_\times$ is the cross product matrix operator.

Considering the constraint of the path we have a vector equality,

$$\ddot{\mathbf{x}} = \mathbf{x}'\ddot{\xi} + \mathbf{x}''\dot{\xi}^2 = m^{-1}\sum_{i\in\mathcal{S}} \mathbf{B}_i\boldsymbol{\phi}_i + \mathbf{g}, \tag{5}$$

Together with (4) this forms the basis of our optimization. Note that linearity in both $\ddot{\xi}$ and $\dot{\xi}^2$, as pointed out in [26], allows optimization problems constrained by this equality to remain convex.

The maximum possible path acceleration at a point $\left(\xi, \ \dot{\xi}\right)$ can be found

$$\begin{aligned} \ddot{\xi}_{\max}(\xi, \dot{\xi}) = &\underset{\boldsymbol{\Phi}_\mathcal{S} \ \mathcal{S}\in\mathfrak{V}_\xi}{\text{maximize}} && \ddot{\xi} \\ &\text{subject to} && \text{inequalities (1) and (2),} \\ & && \text{equalities (5) and (4),} \end{aligned} \tag{6}$$

and will identify the best possible stance for acceleration. Calculation of $\ddot{\xi}_{\min}$ simply converts (6) to a minimization. Note that the two optimizations can and almost always do settle on different stances.

The maximum and minimum feasible speed-squared for any stance $\mathcal{S} \in \mathfrak{V}_\xi$ can be found,

$$\begin{aligned} \dot{\xi}^2_{\mathcal{S},\max}(\xi) = &\underset{\boldsymbol{\Phi}_\mathcal{S}}{\text{maximize}} && \dot{\xi}^2 \\ &\text{subject to} && \text{inequalities (1) and (2),} \\ & && \text{equalities (5) and (4).} \end{aligned} \tag{7}$$

If $\dot{\xi}^2_{\mathcal{S},\max} < 0$ there are no valid speeds. If $\dot{\xi}^2_{\mathcal{S},\min} < 0$, then the minimum speed is zero. The ultimate speed allowed by any valid stance cannot be exceeded, nor can the maximum safe operating speed model parameter ($s_{\min}$) be exceeded. The maximum speed for the purposes of planning is thus the lesser of these two values.

IV. Dynamic Multi-Contact Planning

This paper's application of the path-following multi-contact point mass model to dynamic multi-contact locomotion planing is based on the idea of bang-bang optimal control, and its extension to systems with a speed limit [9].

A bang-bang optimal control strategy solves the problem of minimum time travel given acceleration limits by choosing maximum acceleration from the start and minimum acceleration until the end, with a transition at the unique *switching point* which lets the strategy reach the goal. One straightforward way to calculate this switching point is to simulate maximum acceleration forward from the start point, and then to simulate minimum acceleration backwards in time from the end. Where the two lines intersect in the phase space of position marks the switching point in both position and speed. Points in the phase space which are above the maximum acceleration line are too fast to be reached from the start, and points which are above the minimum acceleration line are too fast to bring to a stop—at least without overshooting the objective and turning around. The area above each line represents a phase space region which is unusable for one of these two reasons, and the union of the two regions represents points which are unusable for either reason. The final path bounds this union from below: it uses the fastest allowable speed for any ξ, and thus has the minimum traversal time over all valid trajectories.

Bang bang optimal controllers are easy to visualize for second order linear systems with constant acceleration bounds, however our system is non-linear and has non-constant acceleration bounds—which depend both on speed and on position. This means that when our system attempts to accelerate, the maximum possible acceleration may be negative—the system may be *forced to decelerate*. The same applies for the maximum deceleration case—it may be impossible to decelerate over some segment of the path, or for some range of speeds.[2] Another alternation from the simple case is that we must explicitly assume that valid trajectories satisfy $\dot{\xi} > 0$. Optimal paths satisfy this naturally in the simple case, but this is no longer true when we consider non-constant acceleration constraints.

This paper proposes a planner for minimum time traversal of a path-following multi-contact point mass system. We treat the planner as a hybrid automata system (Fig. 7) with discrete event boundaries switching between different dynamic behaviors. Each state of the automata, each *automata-state*, has different equations defining its dynamics. The automata-states of the planner system loosely correspond to the phases of the original minimum time trajectory scaling algorithm:

[2]While in the single support phase of a walking gait, for example.

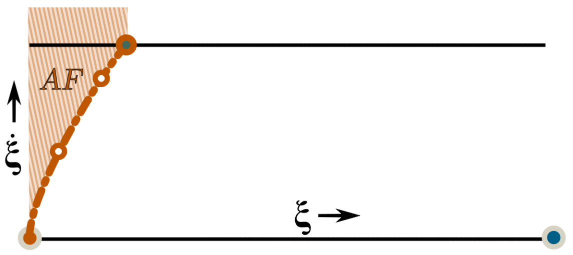

Fig. 3. **First step of the planning process.** The integration of the accelerate forward state AF is represented by ●, and the transition between different stances by ○. The shaded area ▨ represents points which have inaccessibly high speeds, speeds which could only be reached by if the system had started with a speed higher than that of the start point ●.

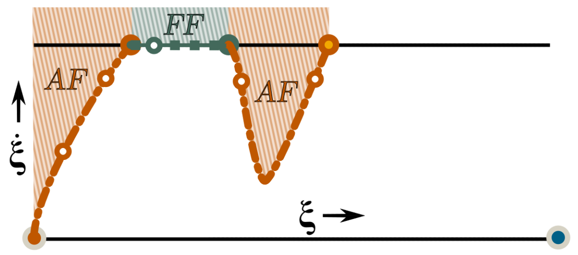

Fig. 4. **Dealing with maximum speed: following maximum speed until forced to decelerate relative to it.** Extends the scenario of Fig. 3. The maximum speed is followed by the forward follower system FF, ●. A stance change internal to the FF system occurs at ○. Shaded area ▨ above the FF system's trajectory is above max speed. The FF system's trajectory is terminated when the system is suddenly forced to decelerate. Because its trajectory is ended by forced deceleration instead of forced acceleration, the FF system transitions back to the AF system.

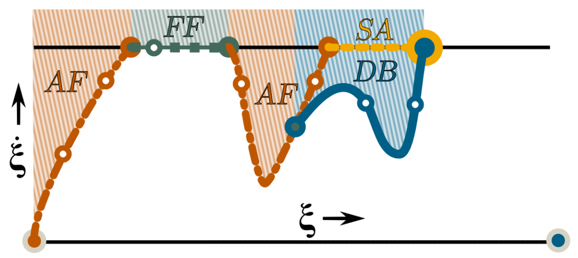

Fig. 5. **Forced acceleration max speed case.** Extends from Fig. 4. Search ahead system SA's trajectory, shown ●, terminates at critical point ●. Deceleration backwards in time system DB's trajectory ● marks the lower bound of a region, shaded ▨, in which speed is too high to avoid going above the maximum speed in the future.

acceleration, deceleration, and search [9]. In the original algorithm the dynamics of acceleration and deceleration were continuous. They were described by one ODE. But to handle multi-contact, each automata-state is now a hybrid system in its own right, with additional *internal state* information representing—in essence—which multi-contact stance is being used. Other cases of internal state are explained later. Event boundaries control transitions between automata-states. Additionally, each automata-state has internal transitions which change its internal state. The planner partitions the final trajectory into segments, and associates an internal state to each segment. The automata-state transition graph is presented in Fig. 7, and Figures 3–6 demonstrate its operation in the phase space.

The states of the automata are labeled AF—"accelerate forward", FF—"follow forward", DB—"decelerate backward", SA—"search ahead", F—"fail", and S—"succeed". Our approach begins with the AF system, which chooses both stances and reaction forces to maximize acceleration at any point in the state space. The AF internal state is often simple: merely the best stance is sufficient. Internal transitions for this automata-state occur when a stance upsets the current best. FF always has the maximum slope in the phase space, and the area above a phase space trajectory segment generated by FF can only be reached by trajectories which had a higher speed for the same ξ as the point where the FF trajectory began. This region is explained in Fig. 3

As shown in Fig. 4, when the AF state reaches a followable maximum speed, the state machine switches to the FF state. Internal changes within this FF automata-state represent a change in the set of stances which are capable of following the maximum speed curve. The FF also represents the case where the maximum speed can only be maintained via high frequency switching between two stances. In this case one stance can only result in acceleration relative to the maximum speed, and the other can only result in relative deceleration. Infinite frequency chatter can be considered a more general case of following the maximum speed curve. A segment of the max speed curve is *followable* if it can be followed at all, and *strictly followable* if it can be followed without infinite frequency chatter. When the FF state reaches a position on the max speed curve where it is no longer followable, it switches to another state. If the system is forced to decelerate relative to the maximum speed curve, then the next state is AF, as Fig. 4 illustrates. Points above the AF state's trajectory are naturally unreachable: they are above the maximum speed.

As illustrated in Figs. 5 and 7, if the forward accelerator AF reaches the maximum speed and cannot begin to follow it, the state machine switches to the search ahead state SA. If the maximum speed was reached, then clearly at least one stance allows acceleration. The system must be in a state of *forced relative acceleration*, where the minimum acceleration the robot is capable of producing, $\ddot{\xi}_{\min}(\xi, \dot{\xi}_{\max})$, forces the robot to exceed the maximum speed. To avoid exceeding the maximum speed the robot must preemptively slow down. The "critical point", [10] and [27], occurs at the transition between forced acceleration and a followable region. It marks the end of the optimal preemptive slowdown trajectory, and the end of the SAstate.

Any phase space point above a minimal acceleration trajectory which ends at a critical point will be forced to exceed maximum speed. By switching to the DB state at a critical point, the state machine finds exactly this trajectory.

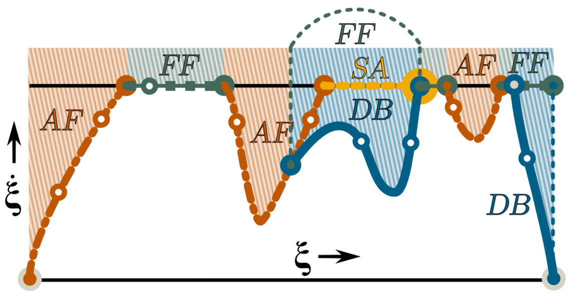

Fig. 6. **Reaching the end.** The *FF* system's jump from the point of intersection to the start of its own trajectory shown thin and dotted, as with the *DB* system's jump to the endpoint ●. This final *DB* trajectory marks the lower bound of the region of points which are too fast to stop by the time they reach the end of the path. Together, the trajectory forming the lower boundary of the union of all shaded regions is the optimal trajectory through the phase space. The regions separated by transitions between trajectories and stance change markers ○, ○, and ○ are allocated to the same stance, list of functionally equivalent stances, or pair of lists of stances—chattering between any selection of one stance from each list.

Edge	condition	jump
FF→*AF*	forced deceleration	
FF→*SA*	forced acceleration	
AF→*FF*	followable max speed	
AF→*SA*	non-followable max speed	
SA→*DB*	critical point ● located	
SA→*DB*	end of path reached	to end ●
DB→*FF*	intersection found	to c.p. ●
DB→*SA*	no intersection	
○, ○, ○	internal state change	
$\star \rightarrow F$	zero or min speed	
DB→ S	final intersection	

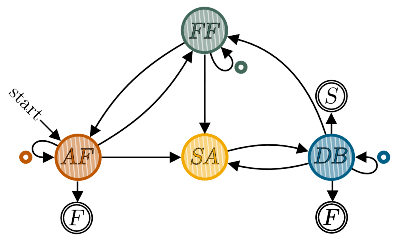

Fig. 7. **Max Speed Planning State Machine with full list of state transitions.** *AF* represents the accelerate forward state *FF* the maximum speed following state (following it forward in time), *DB* the backward minimum acceleration state, and *SA* the search ahead state. S is the "success" state, in which the problem admits a solution. F represents "failure"—a problem with no valid trajectories.

The *DB* state integrates the dynamics of the robot backwards in time, using the minimum possible acceleration. It moves from right to left in the phase space, until it finds an intersection with the previous trajectory segments.

After the intersection is found, the trajectory segments generated by *DB* are reversed. They were generated backwards in time. They are spliced with the forward trajectory segments up until the point of intersection. This forms the new "forward trajectory". The automata, in the *FF* state, then continues forward from the critical point. As shown in Fig. 6, the same procedure is used to end the planning process. The end point is treated as a final critical point.

Ultimately, the union of all excluded regions in the phase space is bounded from below by the final trajectory. Excluded points are too fast to either 1) be achieved from the starting point, 2) avoid crossing the maximum speed curve, or 3) avoid overshooting the end point.

Sometimes the most effective acceleration or deceleration strategy is to temporarily follow the maximum speed curve of a particular stance—a curve below the true maximum speed curve. This can happen if there is a stance with a high speed limit but which forces the robot to decelerate. This situation can even include optimal chatter within the *AF* or *DB* states. When this happens the internal state contains two lists of stances—just like when there is chatter in the *FF* state.

Under this model, not all problems admit a valid trajectory. Untraversable combinations of path and stance list are detected when any state passes below zero speed or the minimum speed of all available stances. A trajectory which reaches minimum speed guarantees that no speed is valid for that ξ, and thus that no phase space path connects the endpoints.

It is possible to guarantee that limbs are not re-allocated to a new location instantaneously. The stance list input to the planning process must guarantee this order itself. That is, for any ξ the valid stances must either uses a limb at its assigned contact, or not use it. By ensuring a ξ region exists in which no valid stance uses a limb, a lifting phase can be guaranteed. Its temporal duration would be at least the ξ distance divided by the maximum speed.

Similarly, chatter in the result could be avoided by only permitting a single stance to be valid for every ξ. However, by restricting the problem in this way many important decisions—which stances to use, and at which point to switch stances—have already been made. With only a single stance for every ξ, the linear program could be expressed in a way compatible with the prior art. Until the stance allocation and chatter problem is solved, the potential of this algorithm lies in its ability to generate *infeasible* trajectories which approximate and bound the feasible ones. Though they have sudden stance changes and chatter, these *limiting trajectories* may serve as the basis for making more difficult decisions, and may offer quick insight on the problem itself. If such complicated behavior arises in the simple case, the behavior is at least as complex with minimum transition times.

V. Some Simple Examples

A simplified Python implementation of the above planning algorithm was constructed, restricted to 2D and using a fairly restrictive maximum speed. The plan was generated first, and then the position trajectory of the center of mass was tracked by a linear controller. This controller follows its own separate force optimization, and is responsible for stabilizing the path-following behavior. The controller shares structure with the planner, and also exploits what we call the repeated linear programming structure, using a remembered active set.

This reduces its complexity. With 1000 iterations, moving slowly along the path in triple contact, the solver takes 15 s without and 0.643 s with the active set remembering alteration. This optimization is much simpler than the one in [28], which also remembers the active set, yet it runs at approximately the same speed (1 ms per solution on a quad core i7). This suggests that the control implementation has a lot of performance improvement still to be realized. Since the planner uses fundamentally the same code, the same is true of the planner. Currently the planning process takes 0.43 s (real-time on an Intel Core i7-3770, 3.40 GHz) to plan the 0.89 s motion shown in Fig. 8. The more complex plan in Fig. 10 takes 0.69 seconds to plan, and has a duration of 1.43 s. Without re-using the active set, this same planning process takes 7.6 s.

The result of planning in a scenario with a ceiling and a gap in the floor is presented in Fig. 8. This experimental setup highlights the dynamic aspect of these plans: it is not possible to traverse this path quasi-statically, because the robot can't balance itself above the gap. This is due, in part, to the available list of stances. There is no stance on the list which is valid above the gap and which permits zero acceleration at zero speed. The robot can choose from only three stances, which are easily distinguishable in the figure. The planner's phase space result is shown in Fig. 9 for the same scenario. As might be expected given such wide initial and final stances, the planner reaches maximum speed quickly, stays at maximum speed for the majority of the maneuver, and then comes to a stop fairly quickly.

VI. Discussion

More than just a single purpose tool, the path-following multi-contact point mass model allows understanding dynamic multi-contact locomotion. It expresses concepts of preemptive deceleration, un-necessary stances, speed-based infeasibility, optimal chattering, and untraversable combinations of path and stances.

The algorithm presented here calculates the limit of robot motion as the foot switching speed approaches instantaneous. This is a more tractable problem than one which includes a cost for switching feet. However, for any algorithm which does include switching cost, our algorithm should be a relaxation.

To include a switching cost would likely sacrifice either optimality or speed. The state space in which we plan is only two dimensional, but to consider this problem we would need to make the current stance into a state, rather than a control input as it is now. True optimal planning in high dimensional state spaces is a nightmare replete with dead-ends and local minima. We suspect that fast, yet approximate reactive algorithms will rely on basic principles such as the limiting case solution we have provided. From the point of view of a robot trying to recover from a large disturbance, there is a time cost to computing. Time spent computing delays the onset of a strategy. A cost function that includes computation time may still have a knowable optimum. As a relaxation of the dynamics, the algorithm presented here could calculate an upper bound on the benefit of continued optimization of a foot-switching trajectory.

When using this algorithm as a heuristic for planning, or as a cost function for kinematic optimization, repeatability is important. So too is the ability to calculate derivatives with respect to the path parameters.

Some may notice that the structure of the optimization problem, including the selection of a stance, can be formulated as a mixed integer problem. If a mixed integer approach could guarantee that some contacts do not appear in the best stance then many stances would not need to be evaluated, and this could save considerable time. There may be merit in relaxing the optimization over multiple stances into one large linear program to eliminate some contacts.

Implicit in the formulation of this model is the idea that speed will strongly influence the choice of contact, and the available acceleration potential. But this is not necessarily the best assumption to make. Speed is very important for robots which verge on leaping between contacts, but for more conservative motions there is another option: evaluating stances based on their worst case performance in a range of speeds. A speed-free, or a finitely many speed ranges, formulation may prove better suited to guaranteeing swing time.

References

[1] J. W. Hall, "Lyapunov: A library for numerically integrating nonlinear dynamical systems." **https://github.com/jackhall/Lyapunov**, 2013.

[2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[3] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Access, IEEE*, vol. 2, pp. 56–77, 2014.

[4] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[5] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[6] T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem," *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 317–342, 2006.

[7] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots," in *Algorithmic foundation of robotics VII*. Springer, 2008, pp. 507–522.

[8] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1325–1349, 2008.

[9] J. E. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The international journal of robotics research*, vol. 4, no. 3, pp. 3–17, 1985.

[10] K. G. Shin and N. D. McKay, "Selection of near-minimum time geometric paths for robotic manipulators," *Automatic Control, IEEE Transactions on*, vol. 31, no. 6, pp. 501–511, 1986.

[11] Z. Shiller and H.-H. Lu, "Computation of path constrained time optimal motions with dynamic singularities," *Journal of dynamic systems, measurement, and control*, vol. 114, no. 1, pp. 34–40, 1992.

[12] Z. Shiller and S. Dubowsky, "On computing the global time-optimal motions of robotic manipulators in the presence of obstacles," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 6, pp. 785–797, 1991.

[13] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *Robotics, IEEE Transactions on*, vol. 30, no. 6, pp. 1533–1540, 2014.

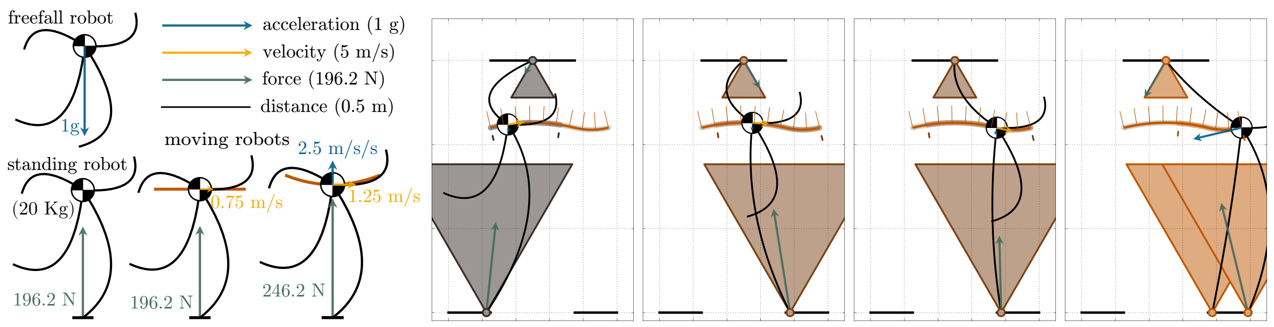

Fig. 8. **Gap hopping video frames.** Selected frames of the attached video. Shaded triangles are color coded to the three different stances available to the planner. Relative scale between forces, distances, speeds, and acceleration is given by legend on the left. Grid at 1/5 meter increments horizontally, 1/2 meter increments vertically. Frames are taken at 0.205 s, 0.405 s, 0.605 s, and 8.885 s, left to right. Plots an animations generated with python's Matplotlib package [29].

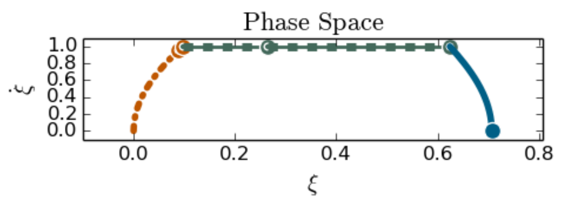

Fig. 9. **Maximum speed planning in the phase space.** Following the style used in the algorithm illustrations, this figure shows acceleration, speed following, and deceleration. Each segment of the plan is linked to the choice of stance used in Fig. 8.

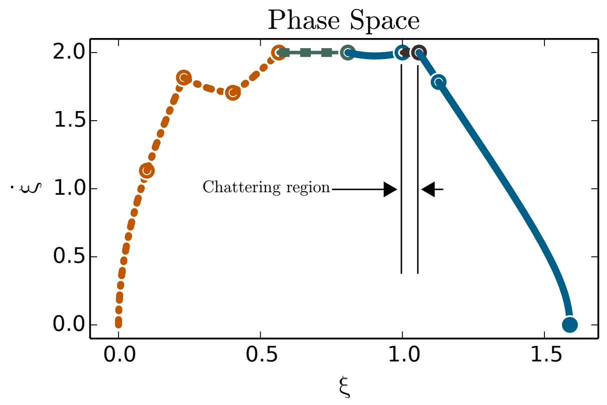

Fig. 10. A more complex maximum speed plan in the phase space demonstrates more of the algorithm: acceleration, following the maximum speed, preemptively decelerating to avoid a region of forced acceleration, a tiny region of chatter, and deceleration to reach the end point.

[14] Q.-C. Pham, S. Caron, and Y. Nakamura, "Kinodynamic planning in the configuration space via velocity interval propagation," *Robotics: Science and System*, 2013.

[15] K. Hauser, "Fast dynamic optimization of robot paths under actuator limits and frictional contact," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2990–2996.

[16] M. Vukobratović and B. Borovac, "Zero-moment pointthirty five years of its life," *International Journal of Humanoid Robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[17] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum model: A simple modeling for a biped walking pattern generation," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1. IEEE, 2001, pp. 239–246.

[18] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. IEEE, 2006, pp. 200–207.

[19] J. Englsberger, C. Ott, and A. Albu-Schaffer, "Three-dimensional bipedal walking control based on divergent component of motion," *Robotics, IEEE Transactions on*, vol. 31, no. 2, pp. 355–368, 2015.

[20] Y. Zhao and L. Sentis, "A three dimensional foot placement planner for locomotion in very rough terrains," in *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*. IEEE, 2012, pp. 726–733.

[21] D. Kim, G. Thomas, and L. Sentis, "Continuous cyclic stepping on 3d point-foot biped robots via constant time to velocity reversal," in *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*. IEEE, 2014, pp. 1637–1643.

[22] ——, "A method for dynamically balancing a point foot robot," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 2015, pp. 901–907.

[23] D. Kim, Y. Zhao, G. Thomas, and L. Sentis, "Assessing whole-body operational space control in a point-foot series elastic biped: Balance on split terrain and undirected walking," *IEEE Transactions on Robotics*, to be published, **arXiv:1501.02855**.

[24] L. Barrios and W.-M. Shen, "Phase space planning and optimization of foot placements in rough planar terrains," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3582–3589.

[25] B. Stephens, "Push recovery control for force-controlled humanoid robots," Ph.D. dissertation, Carnegie Mellon University Pittsburgh, Pennsylvania USA, 2011.

[26] T. Lipp and S. Boyd, "Minimum-time speed optimisation over a fixed path," *International Journal of Control*, vol. 87, no. 6, pp. 1297–1311, 2014.

[27] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 2, pp. 115–123, 1987.

[28] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2589–2594.

[29] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.