An Overview of Optimization-based Motion Planning Methods for Legged Robots: Nonlinear Programs with Fixed Modes and Mixed-integer Quadratic Programs

William Chen
Dept. of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA
verityw@mit.edu

Alex Cuellar
Dept. of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA
alexcuel@mit.edu

***Abstract*—High degree-of-freedom legged robots are increasingly being used in real-world applications due to their ability to traverse complex terrains that traditional wheeled robots cannot. However, this comes at the cost of needing more complex motion planning and control software. We explore two motion planning algorithms on a simple two-legged virtual model of Boston Dynamics' Little Dog robot, which can be generalized to more complex quadruped or humanoid robots. Both leverage numerical optimization tools to account for centroidal physics/dynamics, robot kinematics, and the overall locomotion task to generate actuator commands to allow our robot to walk and run. The methods use a fixed mode (footstep) sequence with non-linear optimization and mixed-integer quadratic programs respectively to incorporate footsteps into the state trajectory search. We investigate and compare the two methods' strengths and weaknesses.**

***Index Terms*—Motion planning, legged robots, quadrupeds, humanoids, nonlinear programs, mixed-integer quadratic programs, optimization**

I. Introduction

As mobile robotics advances, effective and natural control of legged robots has become a cornerstone in the field. In recent years many approaches to the problem of legged robots have been proposed with their own strengths and weaknesses. However, there has always been a struggle between computation power, performance, and generality. The MIT Cheetah for example [1] has extremely high performance, but little generality. Other methods [2] buy generality by sacrificing performance through controlling via a very safe action space.

In this paper, we attempt to study and compare methods on either side of one very small gap along this axis. Specifically, we will focus on two methods for optimizing the the gait limit cycle of a planar version of the Little Dog robot. In one version of the optimization, we use a non-linear formulation with explicitly given the sequence times at which each foot is in contact with the ground (and therefore specifying the patter of a step). In the second, we use MIQP and do not give any a-priori assumptions on the foot-contact sequence during one step. However, this lack of given mode sequence requires tradeoffs in the precision of dynamics. However, we only implement the non-linear method, and only discuss the MIQP method for the purposes of comparison.

II. Related Works

Early models of underactuated robot locomotion designed clever controllers to make a robot act like a simpler and more easy-to-model system. Many such schemes, such as [3] used actuation to model movement as a spring in order to achieve control and attain a stable limit cycle. While extensions of such work remain in the literature [4], advances in optimization techniques have allowed motion models to develop past simulation of basic models and allow for the full range of possibilities with more complex actions.

There are two broad categories of optmimization-based controllers for humanoid walking. First are controllers that attempt to optimize over the full dynamics of the robot as expressed by the full manipulator equation

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = \tau_g(q) + Bu \tag{1}$$

However, most methods using full dynamics require very long computation times and require non-linear solvers [5]. While some methods attempt to linearize these dynamics and perform more fundamental control schemes like LQR, these tend to suffer from issues of robustness [6].

To avoid these issues, others simplify the robot dynamics in order to more easily optimize a trajectory. The most common method for simplifying dynamics is to focus on keeping the Zero Moment Point (ZMP) of the robot above the support polygon of the feet [2]. While such methods can allow the use of convex optimization, there is a tendency for the results to look unnatural and inefficient. Utilizing the contact wrench sum (CWS) as the criteria for stability instead of the support polygon, more diverse actions have been achieved [7]. Furthermore, Dai, Valenzuela, and Tedrake combined the simple dynamics framework of the center of mass with a

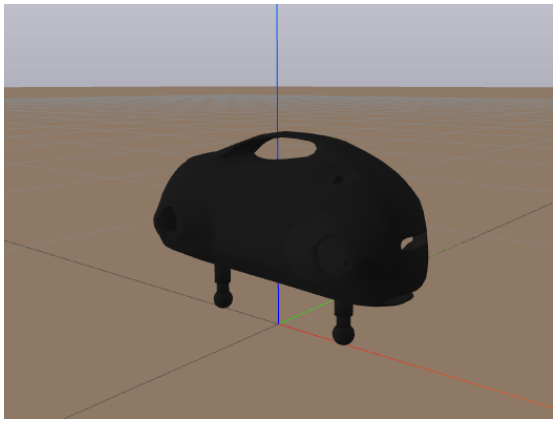

Fig. 1. The planar simplified simulated version of Boston Dynamics' Little Dog robot with two actuators and legs, depicted in its default pose.

full kinematic model [8]. We focus on this method heavily in this paper, with a planar implementation acting as our representative method for a defined step mode-sequence.

Independent of the full/simplified dynamics axis along which to classify controllers, another involves the level of of information given to the optimization about the gait. Many methods of foot placement have been proposed, some of which use sample-based methods such as a modified RRT [9] or search-based methods using A* [10]. Other methods attempt to plan the footstep and robot control in one go utilizing control through contact [11]. However, for this paper, we will focus on a method by Valenzuela that plans footsteps based on CoM dynamics constraints, which are then applied to a nonlinear trajectory planner [12]. By investigating both [8] and [12], we will discuss the gap between planning with a pre-determined mode sequence and planning the trajectory and mode sequence simultaneously.

III. Methodology

A. Problem Specification

We are given a simulated two-legged simplified version of Boston Dynamics' LittleDog robot which only has two actuators connected to two cylindrical legs, located at the front and back centers of the robot (this also means the robot is constrained to the xz-plane, as its actuators cannot provide any roll or yaw, nor any lateral force). The robot's URDF file is shown in the simulation environment in Fig. 1. We wish to find actuator angles that allow the robot to walk forward on flat terrain. We present two optimization-based approaches to achieve this task, with the latter being able to generalize to more complex environments as well.

Both algorithms can generalize to robots with more actuators as well. This is because we primarily look at centroidal dynamics – the movement and rotation of/about the center of mass (COM), drastically simplifying the problem.

B. Nonlinear Program with Fixed Modes

The first method involves posing an optimization problem that integrates the robot kinematics, contact/free-space dynamics, and the task specifications. We formulate the optimization problem in several stages. The code for this implementation is largely adapted from the corresponding LittleDog gait optimization example.

We are given the number of knot points/timesteps in one gait cycle N, the speed we want the robot to move, the stride length, and the stance binary indicator variables $S_i[n]$ foot $i \in \{f, b\}$ is in contact with the ground at timestep n if $S_{ij} = 1$, else the variable is 0. We finally compute $T = \frac{\text{stride length}}{\text{speed}}$, which is the average time it takes for the robot to travel one stride length at the given speed.

We first define some preliminary decision variables. We define $N-1$ decision variables $h_n,\ n \in \{0...N-2\}$ representing the time lengths between consecutive knot points. We then impose the following constraints on them:

$$\begin{aligned} \frac{T}{2} \leq h[n] \leq 2T, \forall n \in \{0...N-2\} \\ \frac{9T}{10} \leq \sum_{n=0}^{N-2} h[n] \leq \frac{11T}{10} \end{aligned} \tag{2}$$

This makes it so that the knot points are not too far apart/too close together and that, in total, they take approximately T total seconds to all occur.

We then define the state and velocity vectors for each of N time steps, we have:

$$\begin{aligned} q[n] = [q_w, q_x, q_y, q_z, x, y, z, \theta_f, \theta_b]^T \ \forall n \in \{0...N-1\} \\ v[n] = [\omega_x, \omega_y, \omega_z, v_x, v_y.v_z, \dot{\theta_f}, \dot{\theta_b}]^T \ \forall n \in \{1...N\} \end{aligned} \tag{3}$$

Where $q_{w,x,y,z}$ represents the robot body's quaternion orientation (meaning we add the unit-length constraint on them), x, y, z represents the translational position, and $\theta_{f,2b}$ are the front and back legs' actuators respectively. Each term of v_n is simply the derivative of the corresponding term in q_n, though note that the first three terms together represent the axis-rotational velocity representation of angular rotation (the magnitude of the vector is the rate of rotation and the direction represents the axis). We also define q_0 to be the robot's default position, which we set to be $[1, 0, 0, 0, 0, 0, 0.1, 0, 0]$ (corresponding to the robot standing upright at the origin, facing the positive x-axis).

We now impose constraints on the joint angles:

$$\begin{aligned} \theta_{f,b}^- \leq \theta_{f,b} \leq \theta_{f,b}^+ \\ \dot{\theta}_{f,b}^- \leq \dot{\theta}_{f,b} \leq \dot{\theta}_{f,b}^+ \end{aligned} \tag{4}$$

Where the superscript $\pm$ represents the upper and lower bounds on joint angles/joint velocities. This constraint is applied for all time.

From there, we define our cost function:

$$\min \sum_{n=0}^{N-1} (q[n] - q_0)^T C_1 (q[n] - q_0) + v[n]^T C_2 v[n] \tag{5}$$

Where $C_{1,2}$ are diagonal positive semidefinite matrices. Specifically, all elements are 0 except for the elements corresponding to $z,\ \theta_{f,b}$ in C_1 and $v_y,\ v_z,\ \dot{\theta}_{f,b}$ in C_2, which are

all 1 (meaning they should not deviate too much from q_0 or get to large respectively).

We then ensure that the velocities in $v[n]$ actually define how $q[n]$ evolve. To do this, note we are given a function $f_v(\dot{q})$, which computes what said velocity vector should look like given the rates of change of the elements in the q vector. Thus, we want:

$$v[n] = f_v\left(\frac{q[n+1]-q[n]}{h[n]}\right) \ \forall n\{0...N-2\} \tag{6}$$

Next, we add contact force decision variables for each of the feet, $CF^{f,b}_{x,y,z}[n] \ \forall n \in \{0...N-2\}$, where the superscript represents which foot the contact force corresponds to. We use a relaxed version of the static friction requirement ($|CF_{x,y}[n]| \leq \mu |CF_z|$) that is linear in our decision variables. Given a coefficient of static friction μ:

$$\begin{aligned} -\mu CF^{f,b}_z[n] \leq CF^{f,b}_{x,y}[n] \leq -\mu CF^{f,b}_z[n] \\ \forall n \in \{0...N-2\} \end{aligned} \tag{7}$$

We also ensure that the normal contact forces are only non-zero when that foot is in stance:

$$0 \leq CF^i_z[n] \leq S_i[n](4mg), \forall n, i \in \{0...N-2\} \times \{f,b\} \tag{8}$$

with total mass m and gravitational acceleration $g \approx 9.81$. Thus, when a foot is not in stance ($S = 0$), then the contact forces are also zero.

Now, we implement constraints so that the variables evolve in accordance to the centroidal dynamics. Starting off, we create decision variables for the position, velocity, and acceleration of the COM:

$$\begin{aligned} COM_{x,y,z}[n], \dot{COM}_{x,y,z}[n], \forall n \in \{0...N-1\} \\ \ddot{COM}_{x,y,z}[n], \forall n \in \{0...N-2\} \end{aligned} \tag{9}$$

As well as angular momentum about the COM:

$$\begin{aligned} H_{x,y,z}[n], \forall n \in \{0...N-1\} \\ \dot{H_{x,y,z}}[n], \forall n \in \{0...N-2\} \\ H_{x,y,z}[n+1] = H_{x,y,z}[n] + h[n]\dot{H_{x,y,z}}[n], \\ \forall n \in \{0...N-2\} \end{aligned} \tag{10}$$

With the latter equivalently being the net torque about the COM.

To define these variables' evolution, we once again make use of three mapping functions: $f_{com}(q, \dot{q} = v)$, $f_H(q, \dot{q} = v)$, $f_{\dot{H}}(q, r, COM, CF)$, which produce the COM position, angular momentum, and net torque, respectively. Setting them equal to the corresponding variables, we have:

$$\begin{aligned} COM_{x,y,z}[n] = f_{com}(q[n], v[n]) \\ \forall n \in \{0...N-1\} \\ H_{x,y,z}[n] = f_H(q[n], v[n], COM_{x,y,z}[n]) \\ \forall n \in \{0...N-1\} \\ \dot{H_{x,y,z}}[n] = f_{\dot{H}}(q[n], r_{f,b}, COM_{x,y,z}[n], CF^{f,b}_{x,y,z}[n]) \\ \forall n \in \{0...N-2\} \end{aligned} \tag{11}$$

Now, all three of those sets of decision variables evolve based on the actual dynamics and kinematics of the robot.

Note that $f_{\dot{H}}$ is nonlinear in the decision variables. In particular, it uses $r^{f,b}_{x,y,z}[n]$ (the position of a given foot at a given time). We can define the function by using the standard net torque around a point (in this case, the COM) formula:

$$f_{\dot{H}} = \sum_{i \in \{f,b\}} (r^{f,b}_{x,y,z}[n] - COM[n]) \times CF^i_{x,y,z}[n] \tag{12}$$

Where $r^i[n] - COM[n]$ is the moment arm around the COM for a given leg. Since gravity acts on the COM, it does not contribute any torque to the overall system, meaning the contact forces are the only ones providing torque.

From here, we can also address static friction at the feet. This is encoded in two constraints that are only implemented when a foot is in stance:

$$\begin{aligned} S_i[n] = 1 \Rightarrow r^i_z[n] = 0 \\ \forall n \in \{0...N-1\}; \\ S_i[n] = 1 \text{ and } S_i[n-1] = 1 \Rightarrow \\ r^i_{x,y,z}[n] = r^i_{x,y,z}[n-1] \\ \forall n \in 1...N-1; \end{aligned} \tag{13}$$

Where the notation ($C \Rightarrow$ constraint) means that the constraint is satisfied if C is true and $i \in \{f,b\}$ indicates which foot is being considered. These constraints mean that if a given foot is in stance, then it is on the ground ($r_z = 0$), and if it was on the ground in the previous time step, it must stay in the same place ($r[n] = r[n-1]$). Recall that r is not a decision variable, but instead a function of the decision variable q (effectively derived from the robot's forward kinematics).

If a foot is NOT in stance, then it has a different constraint applied:

$$S_i[n] = 0 \Rightarrow r^i_z[n] \geq \epsilon \tag{14}$$

for some small value of ϵ. This simply prevents the foot from touching the ground when it is not in stance. By default, we use $\epsilon = .01$.

Finally, we implement task-specific constraints. We wish for our robot to move forward, start at $(x, y) = (0, 0)$, end at (stride length, 0), and not go too low, so we add the following:

$$\begin{aligned} 0 = COM_{x,y}[0] \\ 0 = \dot{COM}_z[0] \\ \text{stride length} = COM_x[N-1] \\ 0.08 \leq COM_z[n] \\ 0 \leq \dot{COM}_x[n] \\ \forall n \in \{0...N-1\} \end{aligned} \tag{15}$$

To extend the derived trajectory, we simply enforce the periodicity constraint:

$$\begin{aligned} q[0] = q[N-1] \\ v[0] = v[N-1] \end{aligned} \tag{16}$$

Note that the first constraint is applied to all elements of

q

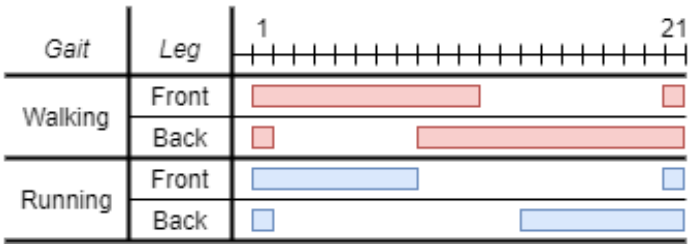

Fig. 2. Visualization of S_i, representing which leg is in stance at each knot point. The colored bars indicate when a leg is in contact with the ground. Red bars are for walking gait while blue bars are for running gait.

except the x element, so that the robot is actually able to move forward.

Thus, in total, as a high level overview of the optimization problem, we have:

$$\begin{aligned} \min \sum_{n=0}^{N-1} (q-q_0)^T C_1 (q-q_0) + v^T C_2 v \\ \text{subject to: centroidal dynamics} \\ \text{robot forward kinematics} \\ \text{legs are in stance at proper times} \\ \text{contact forces/friction applied} \\ N \text{ timesteps over } T \text{ seconds} \\ \text{periodicity constraints} \\ \text{task-specific constraints} \end{aligned} \tag{17}$$

This produces trajectories for the joint angles/velocities and centroidal pose that allow the robot to move forward while meeting the desired mode sequence and obeying the laws of physics.

This optimization problem is implemented in Drake's Python's bindings and solved by the Sparse Nonlinear OPTimizer (SNOPT) solver. We then use Meshcat to visualize our control, state, and pose trajectories.

C. Nonlinear Program with Fixed Modes Results

We test the above formulation with two different mode sequences, corresponding to two different gaits – walking and running respectively.

	Walking	Running
N	21	21
Speed	0.4	0.5
Stride Length	0.2	0.25

Fig. 2 shows the mode sequences for the two legs for each of the two gaits. Note that the running gait has a sequence of knot points where neither foot touches the ground, whereas the walking gait always has at least one foot in stance.

Both gaits succeed in producing the desired behavior in our simulated robot, where the first mode sequence and stride parameters allowed the robot to walk forward, while the second allowed it to run, producing a visible aerial phase (with one such knot frame being visualized in Fig. 3).

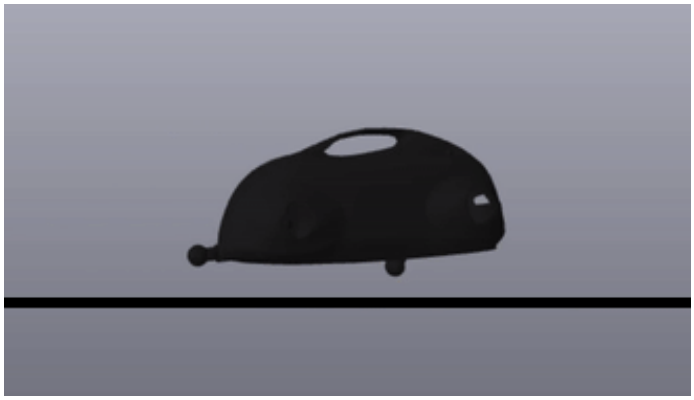

Fig. 3. Example of the aerial phase when the robot is executing the running gait optimization. The horizontal black line indicates where the ground is in the simulator (the ground URDF model was not loaded in, as it interfered with visualizations). Moreover, we increase the value of ϵ to 0.05 in order to better visualize the gap between the foot and the ground in the aerial phase for this image.

IV. Discussion

A. Nonlinear Program with Fixed Modes Discussion

Overall, we succeeded in adapting the NLP motion planning and control for legged robots code to work with a simple planar version of the Little Dog robot. Nevertheless, there are still a few avenues of exploration that can be explored.

The most prominent next step would be to switch over to actually utilizing Drake's contact force simulation. Right now, to visualize the gait patterns, we are just setting the robot state to match what we solved for. That is, we are assuming the dynamics and the like are all perfectly modelled by our nonlinear program, and so if the control commands the program solved for are executed, then the robot would exactly follow the state-space trajectory we found. The robot is not actually being affected by the modelled physics of Drake, but instead just being rendered in the solved poses, akin to a stop-motion animation.

In order to implement this next step, we would need to tune some of the constraints to better match the dynamics of the world. Likewise, we may need to change some of the parameter values – for example, how many time samples there are (as if Drake's simulated time evolution is much more fine than ours', the model of physics encoded in our optimization may not be very accurate). Finally, we would also need to implement some interpolation scheme to get state element values in between key frames (solved for by the optimization). Simple interpolation schema like first-order hold and spherical linear interpolation may be effective for this.

Lastly and obviously, while implementing this would be very time- and resource-intensive, seeing these algorithms being run on a physical legged (non-planar) robot would be extremely instructive.

B. Automating Mode Sequence with MIQP

While the non-linear solution produced good results, the formulation does have limitations. First, non-linear optimization

tends to be somewhat unreliable in real world settings and can fail to find solutions unexpectedly. Second, the computational weight of a non-linear problem makes the addition of mixed integer elements intractable. In this case, mixed integer elements are useful if we want to optimize over the trajectory's mode sequence instead of providing one a piori.

Valenzuela [12] provides a solution to this issue, introducing an Quadratic Program formulation for the dynamics of a planar model Little Dog. Through the more easily solvable framework, Valenzuela is able to introduce optimization over the mode sequence and further narrow the gap between performance and limited hand-designed elements. While the MIQP formulation differs from the one implemented above in several ways, we describe 2 here in depth in order to demonstrate the heart of these methods' differences.

The first necessary change between the non-linear and MIQP methods is approximating the torque as linear constraints. While most CoM dynamics involve linear constraints, the equation relating contact forces to torque is

$$\tau = \sum_{i=0}^{n_{ee}} f_{i,z} s_{i,x} - f_{i,z} s_{i,x} \tag{18}$$

where f_i is the contact force of end effector i and s_i the end effector position with respect to center of mass. Note that this applies to the planar robot only, and a fully 3-dimensional robot would require a cross product to solve for torques. However, since both f and s are decision variables, we cannot use quadratic programming.

To work around these bi-linear terms, Valenzuela uses the McCormick Envelope as a conservative approximation. For the McCormick Envelope, we define a slack variables w defined as $w = uv$ where u and v are the two decision variables of interest. By applying the constraints:

$$\begin{aligned} w &\leq v \\ w &\leq u \\ w &\geq u + v - 1 \\ w, v, u &\in [0, 1] \end{aligned}$$

One can attain a tighter relaxation on the torque constraint by partitioning the the domain over v and u.

While this approximation is ok, can attain tighter approximations with mixed-integer elements. We achieve this by partitioning u into M domains: $0 = u_0 \leq ...u_m \leq u_M = 1$. In this case, our constraints become:

$$\vee_{m=1}^{M} \begin{cases} w \leq u_m^L v \\ w \leq u_m^U v + u - u_m^U \\ u \leq u_m^L v + u - u_m^L \\ w \leq u_m^U v \\ u_m^L \leq u \leq u_m^U \end{cases} \tag{19}$$

Which can be expressed as a mixed integer problem with a convex-hull reformulation. With these multiple partitions and a mixed-integer solver to find the appropriate partition for the desired u and v, we can achieve a significantly tighter relaxation on the dynamics than the vanilla envelope.

The second major difference between Valenzuela's method and the non-linear method implemented above is optimization over a mode sequence. To formulate these constraints, Valenzuela defines different regions $\mathcal{R}_j \forall j \in \{1, ..., n_{regions}\}$ and the forces admissible in the region $\mathcal{K}_j \forall j \in \{1, ..., n_{regions}\}$. While we will not go into as much detail here as the original thesis, both the regions and the forces are convex, and can be combined into a full description of available end effector placements by defining $\mathcal{S}_j = \mathcal{R}_j \times \mathcal{K}_j \forall j \in \{1, ..., n_{regions}\}$. Now, we can formulate the necessary constraints in terms of the end effector placement $\mathbf{r}_{ij}$, forces $\mathbf{f}_{ij}$, and indicator variables z_{ij}, where in each definition i is an end effector and j is a region:

$$(\mathbf{r}_{ij}, \mathbf{f}_{ij}) \in \mathcal{S}_j \forall i \in \{1, ..., n_{ee}\}, j \in \{1, ..., n_{regions}\} \tag{20}$$

$$z_{ij} \in \{0, 1\} \forall i \in \{1, ..., n_{ee}\}, j \in \{1, ..., n_{regions}\} \tag{21}$$

$$\sum_{j=1}^{n_{regions}} \mathbf{r}_{ij} = \mathbf{r}_i \forall i \in \{1, ..., n_{ee}\} \tag{22}$$

$$\sum_{j=1}^{f_{regions}} \mathbf{f}_{ij} = \mathbf{r}_i \forall i \in \{1, ..., n_{ee}\} \tag{23}$$

$$\sum_{j=1}^{f_{regions}} z_{ij} = 1 \forall i \in \{1, ..., n_{ee}\} \tag{24}$$

$$z_{ij} \mathbf{r}_{i,lb} \leq \mathbf{r}_{ij} \leq z_i j \mathbf{r}_{i,ub} \forall i \in \{1, ..., n_{ee}\}, \tag{25}$$

$$j \in \{1, ..., n_{regions}\} \tag{26}$$

$$z_{ij} \mathbf{f}_{i,lb} \leq \mathbf{f}_{ij} \leq z_i j \mathbf{f}_{i,ub} \forall i \in \{1, ..., n_{ee}\}, \tag{27}$$

$$j \in \{1, ..., n_{regions}\} \tag{28}$$

$$\tag{29}$$

Where $\mathbf{r}_{j,lb}$ is the lower bound of region j, $\mathbf{r}_{j,ub}$ the is upper bound of region j, $\mathbf{f}_{j,lb}$ is the lower bound of acceptable force in region j, and $\mathbf{f}_{j,ub}$ is the upper bound of acceptable force in region j. Note that with this formulation, we don't only define the mode sequence, but the position of each end effector at each time step.

From these two major differences, we can clearly see the tradeoff between the non-linear and MIQP method. In exchange for relaxed torque dynamics, we have the capability of more intricate planning across time and spacial regions. Additionally, since Valenzuela's optimization plans foot placement as well as mode sequence, we can apply the MIQP formulation to more complex environments. However, we will not explore such possibilties in this paper.

REFERENCES

[1] Hae-Won Park, Patrick Wensing, and Sangbae Kim. "Online Planning for Autonomous Running Jumps Over Obstacles in High-Speed

Quadrupeds". In: Proceedings of Robotics: Science and Systems. Rome, Italy, July 2015.

[2] Katie Byl et al. "Reliable Dynamic Motions for a Stiff Quadruped". In: Experimental Robotics. Ed. by Oussama Khatib, Vijay Kumar, and George J. Pappas. Springer Tracts in Advanced Robotics 54. Springer Berlin Heidelberg, 2009, pp. 319–328.

[3] Raibert, Marc, Michael Chepponis, and H. B. J. R. Brown. "Running on four legs as though they were one." IEEE Journal on Robotics and Automation 2.2 (1986): 70-82.

[4] Li, Mantian, et al. "Control of a quadruped robot with bionic springy legs in trotting gait." Journal of Bionic Engineering 11.2 (2014): 188-198.

[5] Braun, David J., and Michael Goldfarb. "A control approach for actuated dynamic walking in biped robots." IEEE Transactions on Robotics 25.6 (2009): 1292-1303.

[6] Balancing and Walking Using Full Dynamics LQR Control With Contact Constraints

[7] H Hirukawa, S Hattori, K Harada, S Kajita, K Kaneko, F Kanehiro, K Fujiwara, and M Morisawa. A universal stability criterion of the foot contact of legged robots - Adios ZMP. Proc. of the IEEE Int. Conf. on Robotics and Automation, pages 1976–1983, May 2006.

[8] Dai, Hongkai, Andrés Valenzuela, and Russ Tedrake. "Whole-body motion planning with centroidal dynamics and full kinematics." 2014 IEEE-RAS International Conference on Humanoid Robots. IEEE, 2014.

[9] J. Kuffner et al. "Online footstep planning for humanoid robots". In: IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03. Vol. 1. Sept. 2003, 932–937 vol.1.

[10] L. Baudouin et al. "Real-time replanning using 3D environment for humanoid robot". In: 2011 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids). Oct. 2011, pp. 584–589.

[11] Igor Mordatch, Emanuel Todorov, and Zoran Popović. "Discovery of Complex Behaviors Through Contact-invariant Optimization". In: ACM Trans. Graph. 31.4 (July 2012), 43:1–43:8.

[12] Valenzuela, Andrés Klee. "Mixed-integer convex optimization for planning aggressive motions of legged robots over rough terrain." Diss., Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 2016.