

Dynamics and Domain Randomized Gait Modulation with Bezier Curves for Sim-to-Real Legged Locomotion

Maurice Rahme¹, Ian Abraham², Matthew L. Elwin¹, Todd D. Murphey¹

Abstract—We present a sim-to-real framework that uses dynamics and domain randomized offline reinforcement learning to enhance open-loop gaits for legged robots, allowing them to traverse uneven terrain without sensing foot impacts. Our approach, D²-Randomized Gait Modulation with Bezier Curves (D²-GMBC), uses augmented random search with randomized dynamics and terrain to train, in simulation, a policy that modifies the parameters and output of an open-loop Bezier curve gait generator for quadrupedal robots. The policy, using only inertial measurements, enables the robot to traverse unknown rough terrain, even when the robot’s physical parameters do not match the open-loop model.

We compare the resulting policy to hand-tuned Bezier Curve gaits and to policies trained without randomization, both in simulation and on a real quadrupedal robot. With D²-GMBC, across a variety of experiments on unobserved and unknown uneven terrain, the robot walks significantly farther than with either hand-tuned gaits or gaits learned without domain randomization. Additionally, using D²-GMBC, the robot can walk laterally and rotate while on the rough terrain, even though it was trained only for forward walking.

I. INTRODUCTION

Legged robots can reach areas inaccessible to their wheeled counterparts by stepping over obstacles and shifting their center of mass to avoid falling. However, rough terrain is highly variable and difficult to model: to traverse it, robots must either explicitly sense and adjust or, as in this paper, rely on gaits that are insensitive to ground variation.

We focus on quadrupedal locomotion in unobserved and unmodeled rough terrain. We extend the hybrid learning approach of [1], which uses a learned policy in conjunction with an open-loop dynamics model to control a system. The model provides baseline behavior while the policy uses sensor feedback to adapt the model and improve the control. Specifically, we model leg trajectories using Bezier curve gaits from [2] and augment the underlying gait parameters using a policy trained with reinforcement learning (RL) in simulation. By using dynamics randomization ([3], [4]) while training, the resulting policies produce gaits that, in our

*This work is partly supported by the Northwestern Robotics program. In addition, this material is based upon work supported by the National Science Foundation under Grants CNS 1837515. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the aforementioned institutions.

¹ Department of Mechanical Engineering, Northwestern University, Evanston, IL 60208

² Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213
Corresponding Email: mauricerahme2020@u.northwestern.edu, ia@andrew.cmu.edu

Multimedia can be found at <https://sites.google.com/view/d2gmbc/home>.



Fig. 1. (TOP) Example of sim-to-real transfer of D²-GMBC evaluated on a real robot in difficult terrain. (BOTTOM) A legged robot using an open-loop hand-tuned Bezier curve gait generator failing to traverse the terrain due to brittleness of open-loop gaits.

experiments and without modification, adapt to unmodeled terrain and dynamics.

Although many RL techniques can be used within our framework, we use augmented random search (ARS) [5] for optimizing policy parameters because the resulting policies are linear, enabling efficient implementation on devices with limited computational power.

During training, we provide variable external steering commands to point the robot forward, resulting in a policy that handles steering inputs (from a user or motion planner) without additional training.

The primary contribution of this paper is the Dynamics and Domain Randomized Gait Modulation with Bezier Curves (D²-GMBC) framework, a hybrid approach to legged locomotion combining reinforcement learning and open-loop gaits. With D²-GMBC:

- 1) Policies trained in simulation can be directly deployed on real robots to enable them to walk over rough terrain. We test this sim-to-real transfer experimentally.
- 2) Walking on rough terrain requires only inertial measurements: no foot contact or terrain sensing required.
- 3) We compare D²-GMBC to a similar method without randomized terrain, which serves as a proxy for other methods that randomize dynamics but train on flat

ground (e.g., [6]). D²-GMBC outperforms the non-randomized terrain method in simulated tests over rough terrain, despite performing worse in training.

We also provide plans for the OpenQuadruped (the robot used in our experiments) [7], an open-source simulation environment [8], and additional media at <https://sites.google.com/view/d2gmbc/home>.

II. RELATED WORK

Methods for finding suitable gaits for legged locomotion can be broadly classified into methods that require modeling and tuning [2], [9]–[16], evolutionary approaches [17]–[20], and learning approaches [1], [3], [4], [21], [22].

We focus on three classes of learning approaches here, all of which are trained in simulation and then (mostly) deployed (sometimes with modification) on a robot: modulating gait generators, sim-to-real, and imitation learning.

Modulating Gait Generators: This approach extends existing model-based locomotive gaits by augmenting them via learning. The model-based gait provides a baseline which is improved by a learned residual function. The work of [1] augments sine trajectory gait generators with policies learned in simulation from reinforcement learning. These policies are transferred to real robots to improve forward velocity movement on flat ground. Similarly, [22] uses reinforcement learning on robots to augment elliptical gaits and increase forward speed with three hours of training on a real robot.

We extend these ideas to a broader class of gait generators (i.e., Bezier curves) and situations (rough terrain, lateral and rotational motion). Thus, our approach extends the robot’s capabilities by enabling a human operator or motion planner to provide directional commands.

Sim-to-real: This legged locomotion approach focuses on learning a policy in simulation and transferring it to a real robot without requiring a model-based gait. These ‘sim-to-real’ methods attempt to reduce the gap¹ between learning in simulation and real-world evaluation. The work in [4] stresses the importance of realistic simulations for sim-to-real policy transfer and develops accurate motor representations with simulated noise and latency so that the simulated policies can be used directly on a real robot. Rather than carefully modeling the robot dynamics, the work of [6] shows that policies trained in simulation with randomized dynamic model parameters (e.g., inertia and friction) result in robustness to uncertainty when transferring the policy to a real robot for flat ground locomotion.

Our approach both modulates an existing gait and randomizes parameters to improve the sim-to-real transfer of our policy. We also extend dynamics randomization to domain-randomized terrain to train the robot across a range of terrains that it may encounter in the real world.

Imitation Learning: Imitation learning uses expert data to inform a learning agent. In [23], motion capture data from a dog is adapted for use on a simulated robot using dynamics randomization. To transfer the policy to a real

robot, advantage-weighted regression is used to maximize the reward on the physical system. This method highlights the importance of dynamics randomization and accurate simulation because the reference motions applied directly to the robot result in failure. Our approach is similar to imitation learning in that our expert motions are given by the open-loop stable Bezier curve gait. However, our policies do not require modification to deploy on a real robot.

III. PROBLEM STATEMENT

A. General Formulation

We treat the learning problem as a partially observable Markov decision process (POMDP) where we have a set of uncertain observations of the robot’s state $o_t \in \mathcal{O}$, a reward function $r_t = \mathcal{R}(s_t, a_t)$ defining the quality of the locomotion task as a function of the state $s_t \in \mathcal{S}$, and an action space $a_t \in \mathcal{A}$ containing the set of control inputs to the system. A policy $a_t = \pi(o_t, \theta)$ maps o_t and some parameters θ to a_t , which for us includes inputs to a Bezier curve gait generator and robot foot position residuals.

Given this model, the goal is to find policy parameters θ such that the reward is maximized over a finite time horizon T using only partial observations of the state o_t , according to the following objective

$$\theta^* = \arg \max_{\theta} \mathbb{E} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right], \quad (1)$$

where $0 < \gamma \leq 1$ is a discount factor, θ^* is the optimal policy parameters, T is the episode length (i.e., how long we simulate the robot applying π), and \mathbb{E} is the expected value over the dynamics and domain (D^2) randomized parameters described in Section IV.

In simulation, we have access to the full robot state, which we use to construct the reward function for training; however, full state information is unavailable on the real robot. Therefore, we train our policy in simulation using a partial observation of the state o_t that best mimics the sensors on the real-robot.

B. Reinforcement Learning for Gait Modulation

The problem statement (1) serves as a template for developing and learning policies that adapt open-loop gaits for legged locomotion. During each episode, the policy π is applied to modulate and augment a gait generator. This gait generator (which we describe in more detail in this section) outputs body-relative foot placements which the robot follows using inverse-kinematics and joint control.

Let ℓ be a label for the quadruped’s legs: FL (front-left), FR (front-right), BL (back-left), BR (back-right). An open loop gait is a one-dimensional closed parametric curve $\Gamma_\ell(S(t), \zeta, \beta)$ embedded in \mathbb{R}^3 and specifying the position of a foot in the frame of its corresponding hip. Here, $S(t) : \mathbb{R} \rightarrow [0, 2]$ is a cyclical phase variable, $\zeta \in \mathbb{R}^n$ contains the control inputs and $\beta \in \mathbb{R}^m$ contains the gait parameters (in this paper, $m = 2$). The control inputs ζ and gait parameters β determine the shape of the gait trajectory. The controls

¹This gap is often referred to as the reality gap.

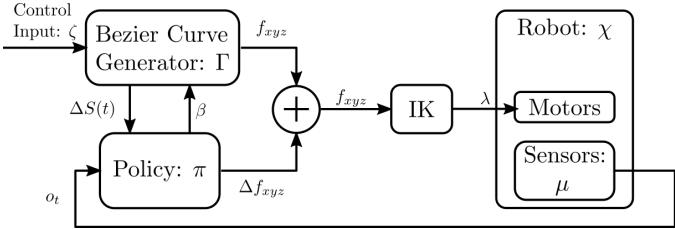


Fig. 2. D^2 -GMBC System Diagram

enable the robot to move in any lateral direction and rotate about its central axis. Control inputs, parameters, and gaits are discussed in Section V.

The goal of the policy

$$(\Delta f_{xyz}, \beta) = \pi(o_t, \theta) \quad (2)$$

is to augment the foot positions output by the gait functions and to modify the open-loop gait generator such that (1) is maximized using only partial observations o_t . Here $\Delta f_{xyz} = [\Delta f_{xyz}^{FL} \ \Delta f_{xyz}^{FR} \ \Delta f_{xyz}^{BL} \ \Delta f_{xyz}^{BR}]$, with $\Delta f_{xyz}^\ell \in \mathbb{R}^3$ and $\beta = \{\psi, \delta\}$, where ψ is the clearance height of the foot above the ground and δ is a virtual ground penetration depth.

The final foot positions are computed as a combination of the gait generator output and the policy residual:

$$f_{xyz} = \Gamma(S(t), \zeta, \beta) + \Delta f_{xyz}, \quad (3)$$

where $f_{xyz} \in \mathbb{R}^{12}$ is the vector of each three-dimensional foot position that the robot should track and Γ contains Γ_ℓ for each leg. The policy both adds a residual term to the output and sets the β parameter. Given the foot positions, the robot computes the inverse kinematics to move its leg joints to the appropriate angles as shown in Fig. 2.

The challenge is to solve (1) in simulation using only the observations o_t such that the resulting policy is suitable for use on a real robot subject to rough terrain and uncertain physical parameters. We provide a solution to this problem using a combination of a simple policy search to solve (1) and dynamics+domain (D^2) randomization in simulation, which we discuss in Section IV. An overview of the gait modulation is provided in Algorithm 1 using the Bezier curve gait generator for Γ which we describe further in Section V.

Algorithm 1 Gait Modulation with Bezier Curves (GMBC)
Given: Policy π with parameter θ , (Bezier) Curve Generator Γ , External motion command ζ , robot sensor observations o_t , Leg phase $S(t)$

- 1: obtain gait modulation from π with learned parameter θ
- 2: $\Delta f_{xyz}, \beta = \pi(o_t, \theta)$
- 3: calculate (Bezier) gait foot placement
- 4: $f_{xyz} = \Gamma(S(t), \zeta, \beta)$
- 5: **return** $f_{xyz} + \Delta f_{xyz}$ to robot for IK joint control

IV. DYNAMICS + DOMAIN (D^2) RANDOMIZATION

We employ two techniques to adapt (1) for improving the performance of sim-to-real transfer of gait modulating policies. Specifically, this approach merges the ideas from [6] to randomize not only the dynamics parameters of the simulated robot, but also the terrain that it traverses. We then solve (1) using a simple policy search method (augmented random search (ARS) [5]) to learn a linear policy as a function of observations subject to sampled variation in the dynamics and domain of the simulated episodes ². We describe the D^2 -randomization below.

Dynamics Randomization We first modify the dynamic parameters that are known to significantly differ between simulation and reality, including the mass of each of the robot's links and the friction between the robot's foot and the ground. The dynamics distribution for which we train the gait modulating policy is $\sigma_{dyn} \sim \mathbb{P}_{dyn}$, where σ is the vector of randomized dynamics parameters which includes the body mass of each link and friction parameter, and \mathbb{P}_{dyn} is a uniform distribution with upper and lower bounds for the mass of each link and friction parameter (see Table I for more detail). At each training epoch, we sample from \mathbb{P}_{dyn} and run training iteration using these fixed sampled dynamics.

TABLE I
DYNAMICS + DOMAIN RANDOMIZATION

Randomized Parameter σ	Range
Base Mass (Gaussian)	1.1kg $\pm 20\%$
Leg Link Masses (Gaussian)	0.15kg $\pm 20\%$
Foot Friction (Uniform)	0.8 to 1.5
XYZ Mesh Magnitude (Uniform)	0m to 0.08m

Domain Randomization In addition to dynamics randomization, we also perform domain randomization by randomize the terrain geometry through which the legged robot is moving (see 3) We parameterize the terrain as a mesh of points sampled from $\sigma_{dom} \sim \mathbb{P}_{dom}$, where σ_{dom} is the variation on the uniform mesh grid, and \mathbb{P}_{dom} is a bounded uniform distribution for which we vary the mesh grid (see Table I for more detail). As with dynamics randomization, we sample the terrain geometry from \mathbb{P}_{dom} and train an iteration of ARS on a fixed sampled terrain.

Let $\sigma \sim \mathbb{P}$ be the joint distribution of \mathbb{P}_{dyn} and \mathbb{P}_{dom} over which we take the expectation in (1), where σ is the joint D^2 sample. The simulation training is described in more detail in Algorithm 2 using ARS and GMBC in Algorithm 1. Policies generated from Algorithm 2 are defined as D^2 -GMBC policies. During training, the external commands ζ are held fixed and defined by the task i.e., move forward at a fixed velocity.

In the next section, we describe the specific Bezier curve gait generator Γ used throughout this work.

²An episode being a single T step run of the simulation with a fixed initial (randomly sampled) state.

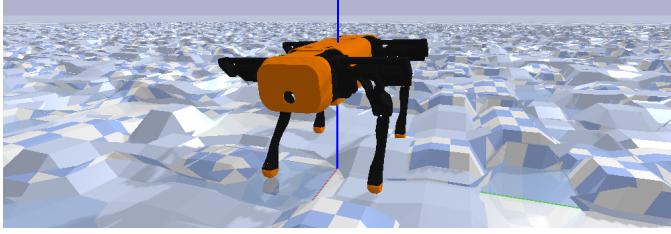


Fig. 3. Illustration of the domain randomized terrain used for training in simulation. The terrain height varies up to 40% of robot height shown in the image [8], [24].

Algorithm 2 RL Simulation training for D²-Randomized GMBC (D²-GMBC) using Augmented Random Search [5]

Initialize: policy parameters θ_0 , D² distribution \mathbb{P} , reward function \mathcal{R} , GMBC (Algorithm 1), iteration number $k = 0$, construct ARS.

```

1: while training not converged do
2:    $\sigma \sim \mathbb{P}$  sample D2 parameters
3:   ARS step of (1) with D2-randomization + GMBC
4:    $\theta_{k+1} \leftarrow \text{ARS}(\pi, \theta_k, \mathcal{R}, \sigma, k)$ 
5:    $k \leftarrow k + 1$ 
6: end while
7: return  $\theta_k$ 
```

V. EXTENDING BEZIER CURVES USING MOTION COMMANDS

D²-GMBC uses an extended and open-loop version of the Bezier curve gaits developed in [2], combining multiple 2D gaits into a single 3D gait that enables transverse, lateral, and rotational motion. Our policy constantly modifies these trajectories, adapting them to uneven terrain while removing the need to sense foot impacts.

A gait trajectory is a closed curve that a foot follows during locomotion. It consists of two phases: swing and stance. During swing, the foot moves through the air to its next position. During stance, the foot contacts the ground and moves the robot using ground reaction forces. A gait is parameterized by a phase $S(t) \in [0, 2)$, which determines the foot's location along the trajectory. The leg is in stance for $S(t) \in [0, 1)$ and in swing for $S(t) \in [1, 2)$.

A trajectory generator $\Upsilon(S(t), \tau)$ maps phase and step length τ to a set of trajectories in \mathbb{R}^2 . We use a trajectory consisting of a Bezier curve during swing and a sinusoidal curve during stance, see Section VII-A for details.

To robot has three control inputs: $\zeta = [\rho \bar{\omega} L_{\text{span}}]$, where $\rho \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ is the trajectory's rotation angle relative to the robot's forward direction, $\bar{\omega}$ is the robot's yaw velocity, and L_{span} is half the stride length.

Locomotion consists of a planar translation $f_{qz}^{\text{tr}} = \Upsilon(S(t), L_{\text{span}})$ and yaw trajectory $f_{qz}^{\text{yaw}} = \Upsilon(S(t), \bar{\omega})$. These trajectories also depend on curve parameters $\beta = [\psi \delta]$ (see Fig. 4 and Section VII-A).

Using the control inputs, we convert the planar trajectories f_{qz}^{tr} and f_{qz}^{yaw} into 3D foot-position trajectories f_{xyz}^{tr} and

$f_{xyz}^{\text{yaw}, \ell}$, where each leg ℓ has the same translational velocity but different yaw velocity. Finally, we transform the yaw and translational curves into a frame relative to each leg's rest position f_{xyz}^{stand} to get the final foot trajectory for leg ℓ :

$$f_{xyz}^{\ell} = f_{xyz}^{\text{tr}} + f_{xyz, \ell}^{\text{yaw}} + f_{xyz}^{\text{stand}}. \quad (4)$$

This scheme enables movements encompassing forward, lateral, and yaw commands and enables policies for straight-line motion to extend to more complex motions.

In particular, let $(f_q^{\text{tr}}, f_z^{\text{tr}})$ and $(f_q^{\text{yaw}}, f_z^{\text{yaw}})$ be the coordinates of $f_{qz}^{\text{tr}} \in \mathbb{R}^2$ and $f_{qz}^{\text{yaw}} \in \mathbb{R}^2$ respectively. Rotating planar trajectory f_{qz}^{tr} by ρ yields the 3D foot trajectory:

$$f_{xyz}^{\text{tr}} = [f_q^{\text{tr}} \cos \rho \quad f_q^{\text{tr}} \sin \rho \quad f_z^{\text{tr}}]. \quad (5)$$

Yaw control is inspired by a four-wheel steered car [25]. To trace a circular path, each foot path's angle must remain tangent to the rotational circle, as shown in Fig. 4.

The yaw command for foot ℓ , $g_{xzy}^{\ell}(t)$, depends on the previous foot position relative to f_{xyz}^{stand} (with $g_{xzy}^{\ell}(0) = 0$):

$$g_{xzy}^{\ell}(t) = f_{xyz}(t - 1) - f_{xyz}^{\text{stand}}. \quad (6)$$

The distance and angle of this step in the xy plane are:

$$g_{\text{mag}}^{\ell} = \sqrt{(g_x^{\ell})^2 + (g_y^{\ell})^2}, \quad (7)$$

$$g_{\text{ang}}^{\ell} = \arctan \frac{g_y^{\ell}}{g_x^{\ell}}, \quad (8)$$

where g_x^{ℓ} and g_y^{ℓ} are the x and y components of g_{xzy}^{ℓ} .

Each leg translates at an angle for a given yaw motion:

$$\phi_{\text{arc}}^{\ell} = g_{\text{ang}}^{\ell} + \phi_{\text{stand}}^{\ell} + \frac{\pi}{2}, \quad (9)$$

$$\phi_{\text{stand}}^{\ell} = \begin{cases} \arctan \frac{f_y^{\text{stand}}}{f_x^{\text{stand}}} & \ell = \text{FR, BL} \\ -\arctan \frac{f_y^{\text{stand}}}{f_x^{\text{stand}}} & \ell = \text{FL, BR} \end{cases} \quad (10)$$

where f_x^{stand} and f_y^{stand} are the x and y components of f_{xyz}^{stand} . The leg rotation angle ϕ_{arc}^{ℓ} provides the final yaw trajectory:

$$f_{xyz, \ell}^{\text{yaw}} = [f_q^{\text{yaw}} \cos \phi_{\text{arc}}^{\ell} \quad f_q^{\text{yaw}} \sin \phi_{\text{arc}}^{\ell} \quad f_z^{\text{yaw}}] \quad (11)$$

Algorithm 3 describes the entire gait generation process, and Fig. 2 shows the Bezier controller system diagram.

Algorithm 3 Bezier Curve Generator Γ - Per Leg ℓ

Inputs: ζ

```

1: Map  $t$  to foot phase  $S_{\ell}(t)$  using (17)
2:  $(f_q^{\text{tr}}, f_z^{\text{tr}}) = \Upsilon(S_{\ell}(t), L_{\text{span}})$ 
3:  $(f_q^{\text{yaw}}, f_z^{\text{yaw}}) = \Upsilon(S_{\ell}(t), \bar{\omega})$ 
4:  $f_{xyz}^{\text{tr}} = [f_q^{\text{tr}} \cos \rho \quad f_q^{\text{tr}} \sin \rho \quad f_z^{\text{tr}}]$ 
5:  $\phi_{\text{arc}}^{\ell} = g_{\text{ang}}^{\ell} + \phi_{\text{stand}}^{\ell} + \frac{\pi}{2}$ 
6:  $f_{xyz, \ell}^{\text{yaw}} = [f_q^{\text{yaw}} \cos \phi_{\text{arc}}^{\ell} \quad f_q^{\text{yaw}} \sin \phi_{\text{arc}}^{\ell} \quad f_z^{\text{yaw}}]$ 
7:  $f_{xyz}^{\ell} = f_{xyz}^{\text{tr}} + f_{xyz, \ell}^{\text{yaw}} + f_{xyz}^{\text{stand}}$ 
8: return  $f_{xyz}^{\ell}$  to the robot for joint actuation
```

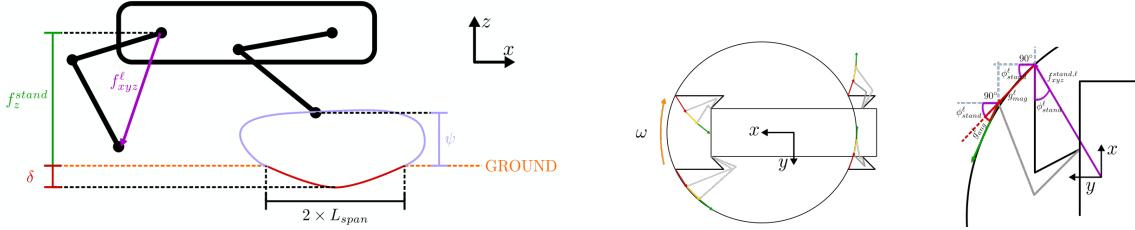


Fig. 4. Schematic of foot placement based on Bezier Gait Generator: f_{xyz} .

VI. RESULTS

We now present several experiments to evaluate our approach, D²-GMBC, in generating legged locomotion for sim-to-real transfer. We first describe the simulated training in more detail, explaining the robot observations, the policy structure, and the reward function. We then evaluate D²-GMBC on the following criteria:

- 1) Generalization to randomized dynamics and terrain.
- 2) Improvement over open-loop gait generators with and without D²-Randomization.
- 3) Sim-to-real transfer performance on a real robot.

A. Simulated Training

We first explicitly describe the simulated training of D²-GMBC. As mentioned in Algorithm 2, we use the augmented random search (ARS) method to train a policy to modulate GMBC (Algorithm 1) using the objective function defined in (1). To match the sensors on the real-robot, we train the policy using an observation comprised of body roll r and pitch p angles relative to the gravity vector, the body 3-axis angular velocity ω , the body 3-axis linear acceleration \dot{v} , and the interleg phase of each foot $S_\ell(t)$, making $o_t = [r, p, \omega, \dot{v}, S(t)]^\top \in \mathbb{R}^{12}$.

We chose a linear policy so that it could run in real-time on inexpensive hardware while improving the open-loop gaits as best as possible. The policy is

$$a_t = \pi(o_t, \theta) = \theta^\top o_t,$$

where $\theta \in \mathbb{R}^{12 \times 14}$. Here, the policy outputs the nominal clearance height and virtual ground penetration depth of the Bezier curve and residual foot displacements that are added to the output of the Bezier curve.

To prevent infeasible foot positions and Bezier curve parameters, we center the policy output within the domain $\pi(o_t, \theta) \in [-1, 1]^{14}$. The output is then remapped to the acceptable range of Bezier curve parameters and a bounded domain of foot residuals. For each of the simulated examples, the high-level motion commands are fixed at $L_{span} = 0.035\text{m}$, and $\rho = 0$. A proportional controller sets the yaw rate $\bar{\omega}$ to ensure that the robot's heading is always zero. Fig. 5 provides an example of a D²-GMBC policy for a single leg.

Augmented random search uses a random parameter search policy optimization technique for reinforcement learning. Thus, for each ARS optimization step, we deploy 16 rollouts per iteration with a parameter learning rate of 0.03 and parameter exploration noise of 0.05. That is, each of the

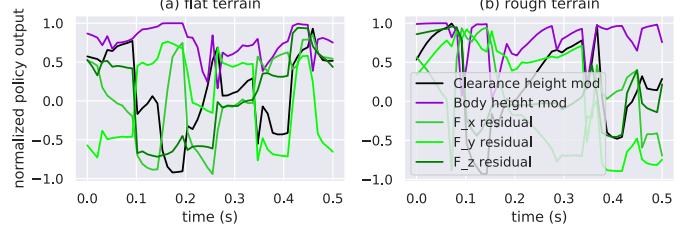


Fig. 5. Example policy output during testing for a single leg. The simple linear policy successfully modulates robot gaits using only on-board inertial measurements.

16 rollouts samples a new parameter $\bar{\theta} = \theta + \Delta\theta$ where $\Delta\theta \sim \mathcal{N}(\mathbf{0}, 0.05)$ where \mathcal{N} is a normal distribution with mean $\mathbf{0} \in \mathbb{R}^{12 \times 14}$ and variance 0.05. Each episode lasts $T = 5000$ steps (50 seconds). The reward function is

$$r_t = \Delta x - 10(|r| + |p|) - 0.03 \sum |\omega| \quad (12)$$

where Δx is the global distance traveled by the robot in the horizontal x -direction in one time step. We found that dividing the final episode reward by the number of time steps improves the policy's learning because it reduces the penalty for a sudden fall after an otherwise successful run, and ultimately encourages survivability. For D²-randomization training, we resample a new set of D² parameters (see Sec. IV) at each training episode of ARS.

B. Improvement to GMBC

To compare the effectiveness of randomizing the robot's dynamics and the terrain for legged locomotion, we train a GMBC policy in simulation with and without D²-randomization and benchmark the policies on unforeseen terrain and dynamics. We then measure how far the robot can travel using each method before it fails. We compare the results of GMBC and D²-GMBC to the open-loop Bezier curve gait generator (unaugmented) by counting the number of times the robot did not fall (exceeded a roll or pitch of 60° or hit the ground) within the allotted simulation time of 50,000 steps, or 500 seconds. We evaluate this over a set of 1000 trials with randomized dynamics and terrain.

As shown in Table II, robots using D²-GMBC have improved survivability on randomized dynamics and terrain when compared to GMBC policies trained on a single set of dynamics and terrain environment parameters; out of 1000 trials, 146 out of 305 (45%) D²-GMBC survivals traveled

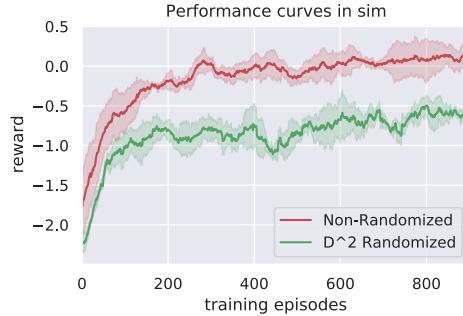


Fig. 6. Evaluated reward curves for simulation training with and without D^2 randomization. Note that despite the improved performance with non-randomization, policies trained with D^2 randomization transfer better to unforeseen dynamics and terrain as shown in Table II.

past 90m, and only 26 out of 327 (8%) GMBC survivals did the same, showing a $5.6 \times$ improvement. Notably, both vastly outperformed the open-loop gait trials, which never made it past 5m and did not survive a single run. The training curves for the D^2 -GMBC and GMBC (non-randomized) policies in Fig. 6 depict that the non-randomized GMBC policy is expected to perform significantly better than D^2 -GMBC policies. These data highlight the potential for simple linear policies to perform well in legged locomotion tasks and provide evidence that training performance is not a useful indicator for testing policy-based locomotion skills in sim-to-real settings.

TABLE II

EACH METHOD WAS TESTED FOR 1000 TRIALS, EACH LASTING 50,000 TIMESTEPS. THE DISTRIBUTION OF MAXIMUM DISTANCES TRAVELED BY THE AGENT AND WHETHER IT FELL OR LIVED ARE REPORTED.

D^2 -GMBC			GMBC			Open-loop	
Distance	# Died	# Lived	# Died	# Lived	# Died	# Lived	
$\leq 5\text{m}$	488	64	450	121	1000	0	
$5\text{m} \text{ to } 90\text{m}$	207	95	222	180	N/A	N/A	
$\geq 90\text{m}$	0	146	1	26	N/A	N/A	

C. Sim-to-Real Transfer

The previous simulations illustrate the generalization capabilities of the linear policy for GMBC using D^2 randomization: training with D^2 -randomization improves legged locomotion performance. We now describe three experiments to demonstrate the sim-to-real transfer capabilities of D^2 -GMBC. We conduct three experiments on OpenQuadruped [7], an inexpensive open-source robot built with hobbyist components.

The first experiment involves testing D^2 -GMBC on a robot whose task is to traverse the 2.2 m track shown in Fig. 7 (a) covered with loose stones whose heights range between 10 mm to 60 mm (roughly 30% of the robot's standing height). The second test evaluates the robot's ability to descend from the peak of loose stones at the maximum 60 mm height onto flat ground shown in Fig. 7 (b). The final test, shows the generalization capabilities of D^2 -GMBC trained in simulation

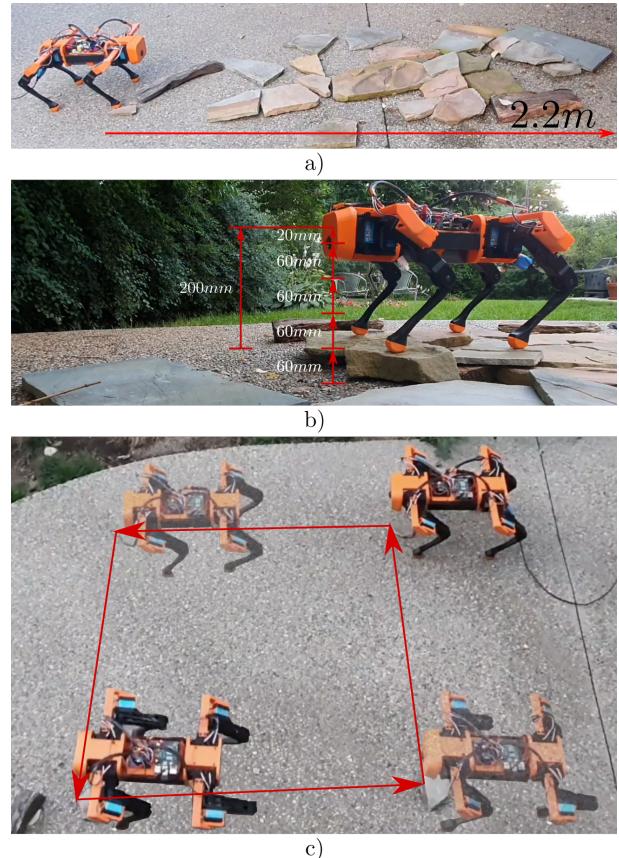


Fig. 7. Illustration of experimental testbeds: a) Experiment 1: Rocky Test Track (2.2m), b) Experiment 2: 60mm Loose Stone Descent, c) Experiment 3: Omnidirectional Performance on Flat Ground. Media available at <https://sites.google.com/view/d2gmbc/home>.

for following unseen high-level motion commands ζ by having the robot follow the 1×1 m square shown in Fig. 7(c).

Experiment 1: Traversing Unknown Terrain In this experiment, we test the policy on terrain that was not seen in training. Additionally, the stones are loose, making the terrain non-stationary and potentially difficult to traverse. Due to the lack of a global odometry, a human operator provided high-level yaw rates to rectify the robot traversing the stones. To prevent human bias, the robot randomly selected with a 50% probability the D^2 -GMBC gait or a benchmark open-loop Bezier curve gait. The experiment continued until both methods were used for at least 10 trials.

TABLE III
EXPERIMENT 1: ROCKY TEST TRACK (2.2M).

	D^2 -GMBC	Open-loop Bezier Gait
Distance Mean (of 2.2)	1.93	1.37
Std. Dev	0.30	0.44
Success Rate (of 1)	0.60	0.14
Distance Improvement	40.73%	
Success Improvement	4.28 \times	

As shown in Table III, we observed a $1.408 \times$ increase in average traversed distance using D^2 -GMBC compared to the open-loop gait. Although the track was only 2.2 m long,

D^2 -GMBC improved the survivability by $4.28\times$ compared to the open-loop Bezier gait. Furthermore, our approach does not require sensing leg joint positions or foot contact.

Experiment 2: Descending from Loose Stones Here, we set the robot’s starting position on a raised platform consisting of 60mm stones to perform a descent test. We record the ratio between successful and failed descents for both D^2 -GMBC and open-loop controllers. The operator, who does not know which policy is being used, drives the robot forward until it either descends or falls. The D^2 -GMBC agent fell 3 out of 11 times and the open-loop controller fell 9 out of 13 times, showing a $2.36\times$ improvement rate on a test with a sharp descent and without contact sensing.

Experiment 3: External Command Generalization This experiment runs the robot on flat terrain to validate the generalization of D^2 -GMBC to external yaw and lateral commands after the sim-to-real transfer and to provide evidence that the improved robustness to rough terrain does not negatively affect performance on flat terrain, even though the policy was trained exclusively for rough terrain.

The operator drove the robot around a $1m \times 1m$ track, performing forward, backward and strafing motions, with some yaw commands to correct the robot’s heading if necessary. Aside from showing the versatility of D^2 -R GMBC in seamlessly providing the operator with mixed motion control, the test allowed us to measure locomotion speed by correlating video timestamps with marks on the ground.

In our tests, the D^2 -GMBC policy, compared to the open-loop gait, was 11.5% faster strafing to the left, 19.1% faster strafing to the right, and had the same forward speed. Backward speed, however, fell by 57.6%. Further inspection revealed that this outlier result was due to the walking stance used to increase robustness causing the two rear motors (both 23kg hobby servos) to reach their torque limits and the rear of the robot to dip. The policy then damped the robot’s motion in anticipation of a fall. Simulation evidence indicates that with more powerful motors, backwards walking would experience no performance degradation.

TABLE IV

EXPERIMENT 3: OMNIDIRECTIONAL SPEED ON FLAT GROUND.

Gait	FWD (m/s)	LEFT (m/s)	BWD (m/s)	RIGHT (m/s)
D^2 -GMBC AVG	0.21	0.29	0.15	0.25
D^2 -GMBC STD	0.04	0.04	0.03	0.05
Open-Loop AVG	0.20	0.26	0.26	0.21
Open-Loop STD	0.02	0.04	0.09	0.04

VII. CONCLUSION

We present a new control architecture (D^2 -GMBC) that augments an open-loop stable gait with a learned policy. Formulating the problem as a POMDP using only IMU data to inform our policy, we generate stable locomotion on unobserved rough terrain. With D^2 -GMBC, policies trained exclusively in simulation with D^2 randomization can be directly deployed on real-robots. The method is data efficient,

achieving a $5.6\times$ high-distance ($\geq 90m$) survival on randomized terrain relative to a similar learning procedure without D^2 randomization in fewer than 600 training epochs. Real robot tests show $4.28\times$ higher survivability and significantly farther travel than a robot using an open-loop gait, despite the terrain being vastly different from the simulated mesh, including having the dynamics of loose stones. On flat ground, omnidirectional speed is preserved. For future work, we are interested to see how this method can be carried over to torque-controllable systems to achieve more dynamic behaviors without terrain sensing.

APPENDIX

A. 2D Bezier Curve Gait

We discuss the Bezier curve trajectories developed in [2].

1) *Trajectory Generation:* Each foot’s trajectory is

$$\Upsilon(S(t), \tau) = \begin{cases} \left[\begin{array}{c} \tau(1 - 2S(t)) \\ \delta \cos \frac{\pi\tau(1-2S(t))}{2\tau} \end{array} \right] & 0 \leq S(t) < 1, \\ \sum_{k=0}^n c_k(\tau, \psi) B_k^n(S(t) - 1) & 1 \leq S(t) < 2 \end{cases}, \quad (13)$$

a closed parametric curve with

$$B_k^n(S(t)) = \binom{n}{k} (1 - S(t))^{(n-k)} S(t). \quad (14)$$

Here $B_k^n(S(t))$ is the Bernstein polynomial [26] of degree n with $n + 1$ control points $c_k(\tau, \psi) \in \mathbb{R}^2$. Our Bezier curves use 12 control points (see Table V). The parameter τ determines the curve’s shape and $0 \leq S(t) \leq 2$ determines the position along the curve. The stance phase is when $0 \leq S(t) \leq 1$ and the swing phase is when $1 \leq S(t) < 2$.

TABLE V
BEZIER CURVE CONTROL POINTS. [2]

Control Point	(q, z)	Control Point	(q, z)
c_0	$(-\tau, 0.0)$	c_7	$(0.0, 1.1\psi)$
c_1	$(-1.4\tau, 0.0)$	c_8, c_9	$(-1.5\tau, 1.1\psi)$
c_2, c_3, c_4	$(-1.5\tau, 0.9\psi)$	c_{10}	$(-1.4\tau, 0.0)$
c_5, c_6	$(0.0, 0.9\psi)$	c_{11}	$(\tau, 0.0)$

2) *Leg Phases:* During locomotion, each foot follows a periodic gait trajectory. The total time for the legs to complete a gait cycle is $T_{\text{stride}} = T_{\text{swing}} + T_{\text{stance}}$, where T_{swing} is the duration of the swing phase and T_{stance} is the duration of the stance phase. We determine T_{swing} empirically (0.2 seconds in our case) and set $T_{\text{stance}} = \frac{2L_{\text{span}}}{v_d}$, where v_d is a fixed step velocity and L_{span} is half of the stride length.

The relative timing between the swing and stance phases of each leg determines which legs are touching the ground and which swing freely at any given time. Different phase lags between each leg correspond to different qualitative locomotion types such as walking, trotting, and galloping. This periodic motion lets us determine the position of every leg relative to $t_{\text{FL}}^{\text{elapsed}}$, the most recent time the front-left leg has impacted the ground. Since our robot does not sense contacts, we reset $t_{\text{FL}}^{\text{elapsed}}$ to 0 every T_{stride} seconds.

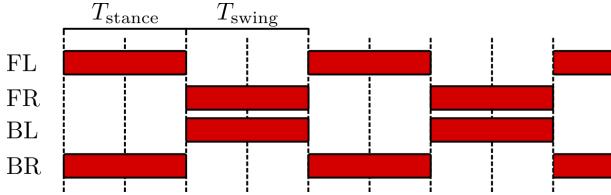


Fig. 8. Leg phases for trotting, where Red is Stance. $T_{\text{stance}} = T_{\text{swing}}$

We define the clock for each leg ℓ to be

$$t_\ell = t_{\text{FL}}^{\text{elapse}} - \Delta S_\ell(t) T_{\text{stride}}, \quad (15)$$

where ΔS_ℓ is the phase lag between the front-left leg and ℓ .

We set the relative phase lag ΔS_ℓ to create a trotting gait:

$$\begin{bmatrix} \Delta S_{FL} \\ \Delta S_{FR} \\ \Delta S_{BL} \\ \Delta S_{BR} \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.5 \\ 0.5 \\ 0.0 \end{bmatrix} \quad (16)$$

The relative timing of the swing and stance phases for such a gait are depicted in Fig. 8.

Next, we normalize the clock for each leg, mapping t_ℓ to the parameter $S_\ell(t)$ such that the leg in stance when $0 \leq S_\ell(t) < 1$ and in swing when $1 \leq S_\ell(t) \leq 2$:

$$S_\ell(t) = \begin{cases} \frac{t_\ell}{T_{\text{stance}}} & 0 < t_\ell < T_{\text{stance}} \\ \frac{t_\ell - T_{\text{stance}}}{T_{\text{swing}}} & -T_{\text{stride}} < t_\ell < -T_{\text{swing}} \\ \frac{T_{\text{swing}} - t_\ell}{T_{\text{swing}}} & -T_{\text{swing}} < t_\ell < 0 \\ \frac{t_\ell - T_{\text{stance}}}{T_{\text{swing}}} & T_{\text{stance}} < t_\ell < T_{\text{stride}} \end{cases} \quad (17)$$

For the first two cases in (17) the legs are in stance and the last two cases they are in swing.

We can then use $S_\ell(t)$ to compute the foot position for leg ℓ using (13). Such a scheme would suffice for forward walking. However, because we need to walk laterally and turn, we employ the techniques used in section V.

ACKNOWLEDGMENT

Thank you to Adham Elarabawy for co-creating and collaborating on the development of OpenQuadruped.

REFERENCES

- [1] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, “Policies modulating trajectory generators,” in *Conference on Robot Learning*, 2018, pp. 916–926.
- [2] D. J. Hyun, S. Seok, J. Lee, and S. Kim, “High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah,” *The International Journal of Robotics Research*, vol. 33, no. 11, pp. 1417–1445, 2014. [Online]. Available: <https://doi.org/10.1177/0278364914532150>
- [3] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami *et al.*, “Emergence of locomotion behaviours in rich environments,” *arXiv preprint arXiv:1707.02286*, 2017.
- [4] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, “Sim-to-real: Learning agile locomotion for quadruped robots,” in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.
- [5] H. Mania, A. Guy, and B. Recht, “Simple random search provides a competitive approach to reinforcement learning,” *arXiv preprint arXiv:1803.07055*, 2018.
- [6] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 1–8.
- [7] M. Rahme and A. Elarabawy, “Open source hobbyist quadruped,” 2020. [Online]. Available: https://github.com/moribots/spot_mini_mini/tree/spot/spot_real
- [8] M. Rahme, I. Abraham, M. Elwin, and T. Murphey, “Spotminimini: Pybullet gym environment for gait modulation with bezier curves,” 2020. [Online]. Available: https://github.com/moribots/spot_mini_mini/tree/spot/spot_real
- [9] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, “Bigdog, the rough-terrain quadruped robot,” *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10822 – 10825, 2008, 17th IFAC World Congress. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1474667016407020>
- [10] I. R. Manchester, U. Mettin, F. Iida, and R. Tedrake, “Stable dynamic walking over uneven terrain,” *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 265–279, 2011. [Online]. Available: <https://doi.org/10.1177/0278364910395339>
- [11] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, “Virtual model control: An intuitive approach for bipedal locomotion,” *The International Journal of Robotics Research*, vol. 20, no. 2, pp. 129–143, 2001.
- [12] D. J. Todd, *Walking machines: an introduction to legged robots*. Springer Science & Business Media, 2013.
- [13] J. E. Shigley, “The mechanics of walking vehicles,” Tech. Rep., September.
- [14] P. A. Bhounsule, J. Cortell, A. Grewal, B. Hendriksen, J. G. D. Karssen, C. Paul, and A. Ruina, “Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge,” *The International Journal of Robotics Research*, vol. 33, no. 10, pp. 1305–1321, 2014. [Online]. Available: <https://doi.org/10.1177/0278364914527485>
- [15] U. Saranli, M. Buehler, and D. E. Koditschek, “Rhex: A simple and highly mobile hexapod robot,” *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 616–631, 2001.
- [16] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, “Control of dynamic gaits for a quadrupedal robot,” in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 3287–3292.
- [17] H. Lipson, J. C. Bongard, V. Zykov, and E. Malone, “Evolutionary robotics for legged machines: From simulation to physical reality.” in *IAS*, 2006, pp. 11–18.
- [18] J. Bongard, V. Zykov, and H. Lipson, “Resilient machines through continuous self-modeling,” *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006. [Online]. Available: <https://science.sciencemag.org/content/314/5802/1118>
- [19] J. Currie, M. Beckerleg, and J. Collins, “Software evolution of a hexapod robot walking gait,” in *2008 15th International Conference on Mechatronics and Machine Vision in Practice*, 2008, pp. 305–310.
- [20] S. Chernova and M. Veloso, “An evolutionary approach to gait learning for four-legged robots,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2562–2567 vol.3.
- [21] J. A. Rieffel, F. J. Valero-Cuevas, and H. Lipson, “Morphological communication: exploiting coupled dynamics in a complex mechanical structure to achieve locomotion,” *Journal of the royal society interface*, vol. 7, no. 45, pp. 613–621, 2010.
- [22] N. Kohl and P. Stone, “Policy gradient reinforcement learning for fast quadrupedal locomotion,” in *IEEE International Conference on Robotics and Automation*, vol. 3. IEEE, 2004, pp. 2619–2624.
- [23] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, “Learning agile robotic locomotion skills by imitating animals,” *arXiv preprint arXiv:2004.00784*, 2020.
- [24] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation in robotics, games and machine learning,” 2017. [Online]. Available: www.pybullet.org
- [25] J. H. Lee and J. H. Park, “Turning control for quadruped robots in trotting on irregular terrain,” in *Proceedings of the 18th International Conference on Circuits Advances in Robotics, Mechatronics and Circuits*, 2014, pp. 303–308.
- [26] G. Lorentz, *Bernstein Polynomials*, ser. AMS Chelsea Publishing Series. Chelsea Publishing Company, 1986.