# Animal Gaits on Quadrupedal Robots Using Motion Matching and Model-Based Control

Dongho Kang, Simon Zimmermann, Stelian Coros

***Abstract*— In this paper, we explore the challenge of generating animal-like walking motions for legged robots. To this end, we propose a versatile and robust control pipeline that combines a state-of-the-art model-based controller with a data-driven technique that is commonly used in computer animation. We demonstrate the efficacy of our control framework on a variety of quadrupedal robots in simulation. We show, in particular, that our approach can automatically reproduce key characteristics of animal motions, including speed-specific gaits, unscripted footfall patterns for nonperiodic motions, and natural small variations in overall body movements.**

## I. Introduction

Today's quadrupedal robots can perform highly dynamic and agile motions: running [1], jumping [2], backflipping [3], and even dancing [4]. Nevertheless, despite their remarkable maneuverability, their movements are arguably stiff and oftentimes lack the subtle nuances that can easily be observed in animals. As an example, during typical locomotion tasks, most quadrupedal robots today tend to follow unnaturally symmetric and regular movement patterns – constant body height, constant forward velocity, military-style trotting gaits, etc. Such motion features can make robots appear stereotypically lifeless and mechanical [5].

Our long-term goal is to make the motions generated by robots appear more natural and organic [6]. To achieve this, we present an approach that combines character animation techniques with a state-of-the-art model-based motion controller architecture. More specifically, the control pipeline we explore is composed of four different stages. First, we adapt data-driven computer animation tools, namely *motion matching* [7]–[9] and *inertialization* [10], to generate an appropriate footfall pattern based on high-level commands such as the robot's desired moving speed and turning rate. Then, we generate kinematic reference trajectories for the base and feet by leveraging motion capture data and the well-known *Raibert heuristic* [11]. These reference trajectories are used as a goal for a model predictive controller which operates on a simplified dynamics model of the robot. Finally, joint-level commands are generated using an inverse dynamics-based whole-body controller.

We experimentally validate our control framework in a simulation environment using the Open Dynamics Engine [13] and apply our technique to various commercially available robot models [12], [14], [15]. We show that without needing precomputation or off-line policy learning, physically simulated robots can generate a variety of periodic and nonperiodic gaits. These gaits and gait transitions emerge naturally as a consequence of user-specified high-level goals. Unlike previous related efforts [16], these behaviors are entirely unscripted.

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 866480).

The authors are with the Computational Robotics Lab in the Institute for Intelligent Interactive Systems (IIIS), ETH Zurich, Switzerland. {kangd, simonzi, scoros}@ethz.ch

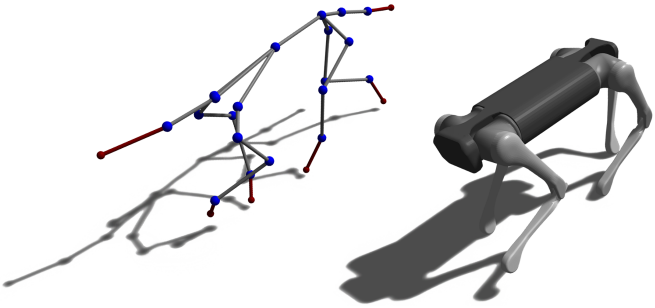

Fig. 1: A dog gait recorded by a motion capture system (**left**), reproduced by the quadrupedal robot *Aliengo* [12] (**right**).

## II. Related Work

As we aim to produce animal-like motions on legged robotic systems, our work lies at the intersection of computer graphics and robotics. Therefore, we introduce a selection of previous work on quadrupedal locomotion control and character animation that are most related to our approach.

### A. Quadrupedal Locomotion Control

Several *model-based control* techniques have been vigorously studied and successfully demonstrated on various quadrupedal robot platforms. One method, namely *Whole-body control* (WBC), finds optimal desired joint-level forces that minimize tracking errors for multiple target accelerations while considering the whole-body dynamics and physical limits of a robot [17]–[20]. This method is dynamically consistent and computationally efficient, but it often fails to perform motion involving frequent non-contact phases. *Model predictive control* (MPC) can resolve this problem by addressing a long-time horizon optimal control. It anticipates the results of the robot's actions and finds an optimal control input for a longer time horizon, at the cost of higher computational complexity [21]–[24]. More recently, combinations of MPC-WBC have been proposed to take benefits of both methods [25], [26]. In this approach, MPC solves a long-time horizon optimal control problem with a simplified model,

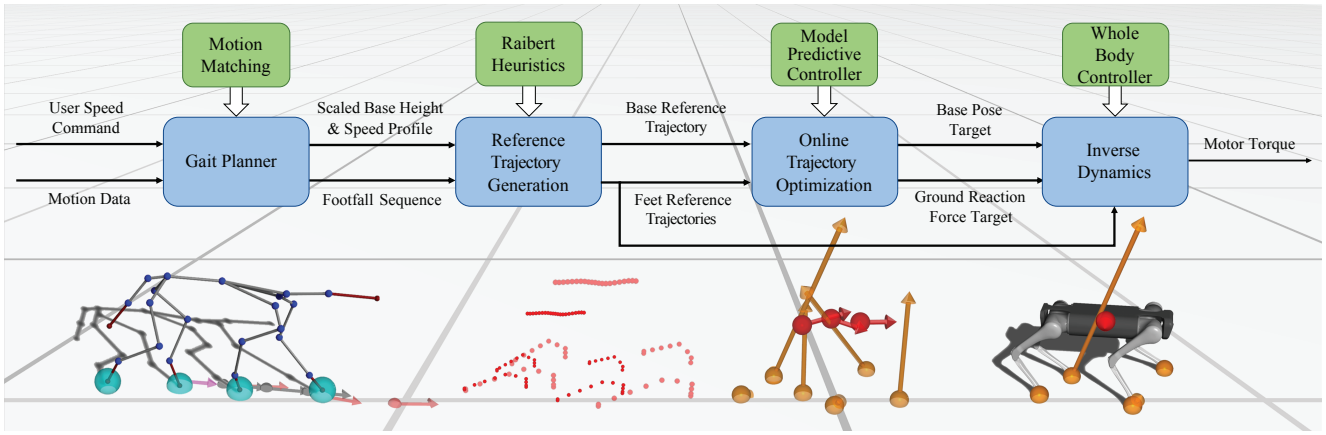


Fig. 2: Control pipeline overview. The individual stages (in **blue**) input and output different parameters (in **black**). The chosen methods are presented in **green**. Each stage is graphically visualized below the block diagram.

and WBC tracks MPC output with a whole-body dynamics model. This hierarchical structure is a simple yet effective solution for versatile and dynamic legged locomotion.

An alternative avenue of model-based control is *learning-based* control, which creates controllers in a data-driven fashion. Several research works propose pipelines to learn a policy for robust quadrupedal locomotion in simulated environments from scratch by applying numerous trial-and-error runs that can later be applied in the real world [27]–[29]. Meanwhile, Peng *et al*. [30] propose a method that utilizes motion capture data of a dog to learn policies that imitate a dog's motor skills.

In this work, we choose a model- over a learning-based approach since our goal is to apply our method to various quadrupedal robots. Model-based control provides more flexibility, as it does not need a time-consuming training procedure for each platform. Specifically, we integrate the combination of MPC-WBC in a similar manner to Kim *et al*. [25]. Our controller is versatile, and can perform various gait patterns without requiring a behavior-specific policy or gain tuning.

### B. Character Animation

In our quest to generate animal-like motions, we turn to the graphics community, which has performed considerable work in this area. In this field of research, it is common to use prerecorded motion capture or image data to generate motions that follow a user's high-level command. Clavet [7] introduces a simple data-driven method called *motion matching*, which generates sequences of character movements from a motion capture dataset. It finds a new motion sequence by searching the entire database for the closest match with respect to predefined features. Alternatively, Zhang *et al*. [31] leverage deep neural networks to directly learn underlying patterns of a motion dataset, then the trained neural network is used to generate new motion sequences for a quadrupedal character. These approaches generate motions without considering physical laws and thus require additional steps for transfer to real systems.

This gap between the kinematic and the physical world has been addressed by *physically based animation* techniques [32]–[34]. The main idea is to learn control policies that track kinematic target motions in a simulated physical environment. Consequently, the resulting motions of virtual agents are not only physically plausible but also have the potential to be transferred to real systems [30]. We note that many physically based animation methods focus on imitating a specific motion clip. In contrast, we aim to build a control pipeline responsive to a user's high-level command. Therefore, we leverage the motion matching technique to generate a target motion on the kinematic level and then transfer it to the physical world as Bergamin *et al*. propose [9]. Our method can seamlessly switch between different gaits based on the input base velocity without a dedicated mechanism. The distinction between their work and ours is that we extract only semantic information from the target motion instead of tracking joint trajectories of the target motion. We use these semantics to create reference trajectories that our model-based controller can track. In so doing, we can transfer animal motions to a robot despite the discrepancy between the animal and the robot's morphology.

## III. Control Pipeline

We present a control pipeline that takes speed commands for the robot's base and a motion dataset as inputs and outputs torque commands for its motors. This pipeline consists of four different stages, as described in Fig. 2. In the following subsections, we discuss the methods chosen for the last three stages. The gait planning method is presented in section IV.

### A. Reference Trajectory Generation

This stage takes inputs from the gait planner and generates reference trajectories for the robot's base and feet. The base trajectory, consisting of six degrees of freedom (three translational and three rotational) per timestep, is constructed given the forward, sideways, and turning speed profile from the previous stage using numerical integration. Meanwhile,

the feet trajectories are generated by linear interpolation between the step locations. These footstep locations for the individual feet are chosen using the robot's kinematics and the *Raibert heuristic* [11]. This well-known tool for legged locomotion ensures that an individual foot lands below the corresponding hip at the middle of the stance phase with the duration of $t_{\text{stance}}$, assuming the robot is moving with constant velocity $\mathbf{v}_{\text{cur}}$. When the robot is required to accelerate to follow a given command velocity $\mathbf{v}_{\text{cmd}}$, the heuristic adjusts the footstep by a feedback term $k_{\text{raibert}}$ that scales the significance of the velocity error. In addition, a centrifugal term is added to generate smoother trajectories under angular velocity command $\boldsymbol{\omega}_{\text{cmd}}$ [25]. In summary, the foot step location $\mathbf{r}_i$ for foot $i$ is

$$\begin{aligned}\mathbf{r}_i \;=\; & \mathbf{p}_{\text{hip},i} + 0.5\, t_{\text{stance}} \mathbf{v}_{\text{cur}} \\ & + k_{\text{raibert}} \left(\mathbf{v}_{\text{cur}} - \mathbf{v}_{\text{cmd}}\right) \\ & + k_{\text{centrifugal}}\, \mathbf{v}_{\text{cur}} \times \boldsymbol{\omega}_{\text{cmd}},\end{aligned} \tag{1}$$

where $\mathbf{p}_{\text{hip},i}$ represents the corresponding hip position in world coordinates. We choose the individual gains as $k_{\text{raibert}} = \sqrt{h/g}$ and $k_{\text{centrifugal}} = 0.5\sqrt{h/g}$ where $h$ is the base height and $g$ is gravitational acceleration.

### B. Online Trajectory Optimization

Since the reference trajectories for the base and feet are generated using kinematic information only, they do not consider any dynamic effects arising from the robot's movements. This can lead to poor performance in tracking the base trajectory and cause the robot to lose its balance and fall over, as the target motion involves frequent underactuated configurations. We therefore need a control stage that matches the input reference trajectories as closely as possible while ensuring that they remain feasible for the robot to track. To achieve this, we apply an MPC scheme. This technique considers a dynamics model of the robot instead of relying on kinematic information only. Furthermore, it predicts the robot's behavior over a long-time horizon, and leads to a more robust control execution that can handle underactuated configurations as well. We observe that animal gaits are typically highly skewed, which is why transferring these gaits to a robot requires a robust control strategy.

We choose the MPC approach presented by Di Carlo *et al*. [23] and Kim *et al*. [25] for our implementation. It approximates the model of the robot into a single lump mass so that its dynamics can be linearized under the following assumptions: first, both roll and pitch angles of the base are small, second, the robot states are always reasonably close to the commanded trajectory. With the linear model, the MPC finds an optimal base trajectory $\mathbf{x}$ and the ground reaction forces of the feet $\mathbf{f}$ for a given discrete time horizon $m$. This problem is formulated as a convex quadratic program (QP) where the global optimal solution can be found efficiently. Note that this MPC formulation does not solve for foot trajectories since optimization over foot placement in combination with ground reaction forces results in a non-convex problem [21], [22], [35] which slows down the computation significantly and tends to get stuck in an undesirable local minimum.

The described MPC formulation can be written as

$$\min_{\mathbf{x},\mathbf{f}} \quad \sum_{i=0}^{m} \left\|\mathbf{x}(k+1) - \mathbf{x}^{\text{ref}}(k+1)\right\|_{\mathbf{Q}} + \|\mathbf{f}(k)\|_{\mathbf{R}} \tag{2a}$$

$$\text{s.t.} \quad \mathbf{x}(k+1) = \mathbf{A}_k \mathbf{x}(k) + \mathbf{B}_k \mathbf{f}(k) + \hat{\mathbf{g}} \tag{2b}$$

$$|f_x| \le \mu f_z, \quad |f_y| \le \mu f_z, \quad f_z > 0 \tag{2c}$$

$$\mathbf{f}_{min} \le \mathbf{f} \le \mathbf{f}_{max} \tag{2d}$$

$$\boldsymbol{\phi}_{min} \le \boldsymbol{\phi} \le \boldsymbol{\phi}_{max}, \quad \boldsymbol{\theta}_{min} \le \boldsymbol{\theta} \le \boldsymbol{\theta}_{max} \tag{2e}$$

where we optimize for a discrete time horizon of $m$ steps. The objective (2a) is to match the base trajectory $\mathbf{x}$ with its reference and keep the ground reaction forces $\mathbf{f}$ small. The linearized lump mass dynamics are included as a constraint in (2b). Furthermore, the Coulomb friction cones for all the feet in contact with the ground are modeled in (2c) and limited by (2d). We ensure that the assumption of small roll $\phi$ and pitch $\theta$ angles is not violated by including the constraint (2e).

After finding the optimal ground reaction force and base trajectory predictions, we linearly interpolate the base trajectory while assuming that the ground reaction force remains constant within one timestep $\Delta t_{\text{MPC}}$. The resulting base trajectory $\mathbf{x}^{\text{MPC}}$ and ground reaction forces $\mathbf{f}^{\text{MPC}}$ are used as targets for the whole-body controller we describe in section III-C.

### C. Inverse Dynamics

As a final step of the control pipeline, we generate joint torque commands by solving an inverse dynamics problem. In this stage, a robot controller needs to fulfill the requirement of running at high-frequency rates (typically over 200 Hz) to allow the robot to perform dynamic maneuvers. In addition, it needs to take into account the whole-body dynamics of the robot to ensure high accuracy. One control method that meets these requirements is *whole-body control (WBC)* [17]–[20].

Our WBC is formulated as a convex quadratic program to allow short computation times. The most noticeable difference to equation (2) is the dynamics model which is replaced by the whole-body dynamics of the robot. Additionally, the problem only has to be solved for one control timestep $\Delta t_{\text{WBC}}$. The whole-body dynamics model is parameterized using generalized accelerations, joint torques and ground reaction forces, which we denote in the following by $\ddot{\mathbf{q}}$, $\boldsymbol{\tau}$ and $\mathbf{f}$, respectively.

$$\min_{\ddot{\mathbf{q}},\boldsymbol{\tau},\mathbf{f}} \quad \left\|\hat{\mathbf{a}}(\ddot{\mathbf{q}}) - \hat{\mathbf{a}}^{\text{cmd}}\right\|_{\mathbf{Q}_1} + \left\|\mathbf{f} - \mathbf{f}^{\text{MPC}}\right\|_{\mathbf{Q}_2} \tag{3a}$$

$$\text{s.t.} \quad \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \mathbf{S}^T\boldsymbol{\tau} + \mathbf{J}^T\mathbf{f} \tag{3b}$$

$$|f_x| \le \mu f_z, \quad |f_y| \le \mu f_z, \quad f_z > 0 \tag{3c}$$

$$\mathbf{f}_{min} \le \mathbf{f} \le \mathbf{f}_{max} \tag{3d}$$

$$\boldsymbol{\tau}_{min} \le \boldsymbol{\tau} \le \boldsymbol{\tau}_{max}. \tag{3e}$$

The optimization variables include $\ddot{\mathbf{q}}$, $\boldsymbol{\tau}$ and $\mathbf{f}$. The equality constraint (3b) is the equation of motion of the whole-body dynamics model. We reintroduce the force limits in (3c) and (3d). Additionally, we add joint torque limit constraints based on the robot's physical limitations with (3e). The objective (3a) consists of two quadratic penalties. The first term matches accelerations of the base and the feet with target acceleration commands, and the second one matches ground reaction forces with those we obtained from the MPC stage. The acceleration vector

$$\hat{\mathbf{a}} = [\mathbf{a}_{\text{base}}, \mathbf{a}_{\text{foot},1}, ..., \mathbf{a}_{\text{foot},n_e}] \in \mathbb{R}^{n_e \times 3+6} \tag{4}$$

is a concatenated vector of the base acceleration $\mathbf{a}_{\text{base}} \in \mathbb{R}^6$ and $n_e$ feet accelerations $\mathbf{a}_{\text{foot},i} \in \mathbb{R}^3$. $\mathbf{a}_{\text{base}}$ is the first six elements of generalized acceleration $\ddot{\mathbf{q}}$, and the acceleration of foot $i$ is computed by $\mathbf{a}_{\text{foot},i} = \dot{\mathbf{J}}_i\dot{\mathbf{q}} + \mathbf{J}_i\ddot{\mathbf{q}}$, where $\mathbf{J}_i$ is a Jacobian matrix of foot $i$.

The target acceleration commands are computed using *implicit PD control*, which can be written as

$$\mathbf{a}^{\text{cmd}} = -\frac{k_p(\mathbf{p} - \mathbf{p}^{\text{target}}) + \Delta t k_d(\dot{\mathbf{p}} - \dot{\mathbf{p}}^{\text{target}}) + \Delta t k_p \dot{\mathbf{p}}}{1 + \Delta t^2 k_p + \Delta t k_d}, \tag{5}$$

where $\Delta t$ is the control timestep size of WBC, $\mathbf{p}^{\text{target}}$ and $\dot{\mathbf{p}}^{\text{target}}$ are the pose and velocity targets respectively. For the base, the targets are generated from the base trajectory prediction $\mathbf{x}^{\text{MPC}}$ by MPC. For the feet, the targets come from the reference foot trajectories from the second stage described in section III-A. The latter is reused under the assumption that the base pose sequence generated by MPC is reasonably close to the previously generated reference base trajectory. Therefore, recreating the foot trajectories using the foot placement rule, equation (1), can be avoided. As a final result of this stage, we find torque commands $\boldsymbol{\tau}^\star$ that minimize the tracking error and send them to the robot.

## IV. Gait Planning using Motion Matching

This section presents our gait planning strategy based on a simple data-driven method called *motion matching* [7]–[9]. We integrate it into our legged robot control pipeline as the key component for generating biological motions for quadrupedal robots. As depicted in Fig. 2, the planner takes a user's target speed command (forward, sideways, and turning) and motion data of real animals as inputs, and outputs semantic information that encodes key characteristics of an animal's motion. More precisely, we extract a footfall sequence, a base height profile, and a base speed profile. A base height and a speed profile entail an animal's inconsistent moving speed as well as light up-and-down and left-and-right body offsets during a gait cycle. Furthermore, a footfall sequence defined by timings of steppings and durations of swings can effectively encode the aperiodicity and asymmetry of animal gaits. These components are then passed on to the next stage of the control pipeline and embedded into reference trajectories (see section III-A). We will now discuss the individual aspects of this procedure in more detail. An overview of the individual steps is given in Fig. 3.

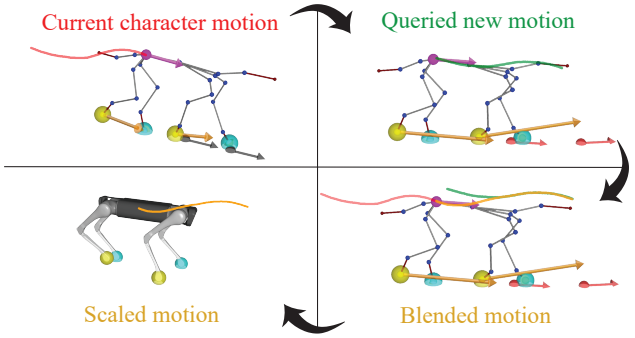


Fig. 3: Gait planning consists of several steps. Using the current character motion and the user input (**top left**), a new motion sequence is queried from the database (**top right**). To ensure a smooth transition, inertialization blending is applied (**bottom right**). Afterwards, we extract footfall sequence and base height/speed profile from blended motion. The base height/speed profile is scaled for the robot (**bottom left**).

### A. Motion Matching

The main idea of motion matching is the following: given the current state of a character and a user's target input, it finds the best matching motion sequence from the dataset to playback. We will now formulate this procedure for our use case. Imagine we have a raw dataset consisting of $n_c$ individual motion clips. We use this to build a list of reference motions for each individual motion clip $Y_i = [\mathbf{y}_1, ...\mathbf{y}_{n_{f,i}}], \forall i \in 1, ..., n_c$, where $n_{f,i}$ denotes the number of frames for motion clip $i$, and $\mathbf{y}_j$ is the state of the character at the $j$th frame. We define the character's state using generalized coordinates and velocities $\mathbf{q}$ and $\dot{\mathbf{q}}$, and therefore $\mathbf{y}_j = [\mathbf{q}, \dot{\mathbf{q}}]$. Note that if the dataset contains pose data only, we compute the generalized velocities $\dot{\mathbf{q}}$ using the finite-difference method.

From $\mathbf{y}_j$, we extract the corresponding *motion feature vector* $\mathbf{z}_j$ as described in [8] but modify the definition for quadrupedal characters as

$$\mathbf{z}_j = [\tilde{\mathbf{p}}_{20}, \tilde{\mathbf{p}}_{40}, \tilde{\mathbf{p}}_{60}, \tilde{\mathbf{h}}_{20}, \tilde{\mathbf{h}}_{40}, \tilde{\mathbf{h}}_{60}, \mathbf{r}, \dot{\mathbf{r}}, \dot{\mathbf{p}}, \phi] \in \mathbb{R}^{43}, \tag{6}$$

where $\tilde{\mathbf{p}}_{20}$, $\tilde{\mathbf{p}}_{40}$, $\tilde{\mathbf{p}}_{60} \in \mathbb{R}^2$ are future ground-projected positions of the character after 20, 40, 60 frames, $\tilde{\mathbf{h}}_{20}$, $\tilde{\mathbf{h}}_{40}$, $\tilde{\mathbf{h}}_{60} \in \mathbb{R}^2$ are future ground-projected heading vectors of the character, $\mathbf{r} \in \mathbb{R}^{12}$ and $\dot{\mathbf{r}} \in \mathbb{R}^{12}$ are positions and velocities of the feet, $\dot{\mathbf{p}} \in \mathbb{R}^3$ is the base velocity, and $\phi \in \{0, 1\}^4$ is the contact state of all four feet, where 0 and 1 indicate swing and stance, respectively. All the position and vector entities are expressed in the character's coordinate frame.

We use this information to build a set of motion feature vectors $Z_i = \{\mathbf{z}_1, ..., \mathbf{z}_{n_{f,i}-61}\}$ for each motion clip $i$. As we use frame $j$ to $j+60$ to build $\mathbf{z}_j$, we receive $n_{f,i}-61$ feature vectors. After repeating this process for the entire dataset, we end up with two databases: an *animation database* $Y$, which is a set of lists $Y_i$, and a *matching database* $Z = \bigcup_{i=1}^{n_c} Z_i$, which denotes a union set of $Z_i$. As a final step, we normalize each feature vector by the mean and the standard deviation of

the entire database. This allows us to use euclidean distances for comparing individual feature vectors.

At runtime, for every $N$ frames (defined in Table I) we build a *query vector* $\hat{\mathbf{z}}$ from the current state of the character and the user command. The future ground-projected position and heading vectors of $\hat{\mathbf{z}}$ are computed by numerical integration of a user command. The query vector is used to retrieve the best matching feature vector $\mathbf{z}^\star$ in $Z$ in terms of minimizing the euclidean distance with respect to $\hat{\mathbf{z}}$. Based on $\mathbf{z}^\star$, the corresponding character state $\mathbf{y}^\star$ and its $N_h$ subsequent states can be retrieved from $Y$. Note that it is not desirable to perform this process for every planning timestep since it may cause cycling over the same motion and increase the computational burden [9]. We use $N = 30$ that corresponds to $0.5\,\text{sec}$.

### B. Inertialization Blending

The new motion sequence found from the database does not necessarily tie in smoothly with the character's currently executed movements. In cases where we have to stitch two nonconsecutive motion sequences together, the transition between the previously played and the new sequence can be significantly discontinuous. Such discontinuity can cause a glitch in the overall target motion. As a result, the extracted information can be unsuitable for the control pipeline, as it most often leads to unstable robot locomotion. To resolve this problem, the transition between the old and the new sequence needs to be smoothed out. Once more, we rely on an established blending tool from computer graphics called *inertialization* [10]. The basic idea behind this technique is that it transitions the character's movements from one sequence to another by interpolating its base and joint states using a fifth-order polynomial [36]. We have found this approach suitable for our application as well, as it results in smooth base speed and height profiles while generating suitable footfall sequences.

### C. Transfer to Robot

After finding a suitable motion sequence for a quadrupedal character, we need an additional step to transfer this motion to a robot. This is because robots typically have a different morphology and actuation power than animals. Therefore, we extract only semantics of the character motions, namely a base speed and a height profile, as well as a foot fall sequence. This choice allows us to use simple scaling on speed and height profiles according to dimensions of the robot. Choosing a suitable scale factor may involve many properties to be considered, such as limb lengths, body dimensions, mass properties, and actuation power. Instead of building these relations, we consider this factor a tuning parameter, as we found it to be intuitive and easy to find.

After applying the scaling onto the currently selected motion sequence, the extracted base speed profiles are further processed by a Gaussian and median filter for noise reduction. In our implementation, we use a Gaussian filter (filter width 7) for forward and sideways speed and a median filter (filter width 5) for turning speed. We observe that this simplifies the tracking of a subsequently generated reference trajectory for the rest of the control pipeline.

A footfall sequence can be described as a sequence of pairs of start and end times of a leg swing. In other words, it describes when a specific leg is in contact with the ground and when it is in swing. As the motion capture data consists of sequences of base and joint poses, the footfall pattern needs to be extracted by thresholding the height and velocity of the character's feet. More formally, we consider a foot to be in swing mode if

$$\|\mathbf{v}_{\text{foot}}\|_2 > \theta_{\text{velocity}} \quad \text{and} \quad z_{\text{foot}} > \theta_{\text{height}}, \tag{7}$$

where the specified thresholds for height and velocity $\theta_{\text{height}}$ and $\theta_{\text{velocity}}$, respectively, can be found in Table I. We note that drift in position and velocity often happens due to noise in the motion data or *foot-skate effects* caused by the inertialization blending [37], which can result in faulty swing detection. However, we found that in practice, the consequences are negligible for our application unless the motion sequence contains highly dynamic maneuvers like quick turning, hopping, or galloping.

As an additional measure, we include a simple heuristic to increase the robustness of the control pipeline by introducing an artificial time shift $t_{\text{shift}}$ (see Table I). Specifically, we generate a gait plan for time $t + t_{\text{shift}}$ at control time $t$. It ensures that our controller does not abruptly switch between a currently executed and a newly found motion. This ultimately improves the collaboration between the different stages, as the locomotion controller is not suddenly interrupted by the motion matching procedure.

## V. Results

We evaluate the efficacy of our control pipeline in simulation using the Open Dynamics Engine [13] and three different robotic platforms from *Unitree Robotics*: *A1* [14], *Aliengo*, [12] and *Laikago* [15] are depicted in Fig. 4. We note that the only parameter that needs to be tuned for a specific robot model is the scaling factor that is used to transfer the character motions from the dataset to the robot (as explained in section IV-C). The experiments are conducted using an Intel i7-9700K CPU, and the required parameters are summarized in Table I. We used a motion capture dataset of a dog available online [31]. As some sequences of the dataset contain motions that are clearly not feasible to reproduce on robots, such as sitting on the ground, jumping, or cantering, we carefully selected $65{,}498$ frames out of $265{,}994$ frames. In the following, we will present our findings using some representative motion sequences. We refer the reader to the accompanying video* for more detailed visualization of the results.

The setup for all conducted experiments was the same: A user can first select the preferred robot model and then introduce real-time target commands for a robot to follow using a simple GUI. These target commands are *forward*, *sideways* and *turning* velocity, expressed in a robot's base

*The video is available in `https://youtu.be/6-zTPTL0fJY`.

coordinate frame. The overall goal is that a robot adapts to a dog's natural-looking motions while following the target commands as closely as possible.

Fig. 5 shows a target motion generated by motion matching and its reproduced version on the Aliengo robot given a constant forward speed command. The robot started from a standing configuration and was then given a target forward speed of $0.8\,\mathrm{m/sec}$. By observing the body trajectories of the robot (middle, in purple), we can see that the robot is able to replicate the small up-and-down and left-and-right body motions that are typically found in quadrupedal animal movements. As visualized in the supplementary video, this behavior heavily contributes to making the robot's movements look more lively. Furthermore, Fig. 5 shows the footfall sequence that the robot is following. It is nonperiodic and involves a smooth and natural transition from a *walk* to a *pace* gait [38]. Both these properties stand in contrast to choosing a fixed, symmetric walking pattern. The visual differences between these behaviors are highlighted in the supplementary video.

Fig. 6 depicts a different motion sequence, where the forward speed command was varied over a range of $[0, 0.9]$ $\mathrm{m/sec}$, and the robot started again from a standing configuration. Fig. 6(a) shows the velocity target command (in blue) versus the base velocity resulting from motion matching (in red) and the robot (in green), where a moving average filter (with filter width $60$) has been used for better visualization. The plot shows that the robot is able to track the inputs coming from the gait planner well (the overall root mean squared error is $0.0656\,\mathrm{m/sec}$). However, there are some mismatches and delays in comparison to the target command. This can be explained by the fact that there is a limited amount of data available. In other words, there is not necessarily a recorded motion available for every possible input velocity command. Despite the local mismatches in base velocity, Fig. 6(c) shows that overall, the target base position resulting from the velocity command can still be tracked reasonably well. We visualize the raw velocity data (i.e. without the moving average filter) in Fig. 6(b). It shows that animals typically do not move with constant body speed (in yellow) but rather travel more flexibly. This is the desired behavior for our application, as it leads to less rigid and more biological motions. The curve in light green shows that the robot can replicate this behavior well. Additionally, the footfall sequence of the robot is depicted in Fig. 6(d). It is visible that the robot swiftly changes its gait based on the target base speed. In the range of $[0, 0.3]$ $\mathrm{m/sec}$, the gait planner generates a slow walking gait that moves the robot's feet one by one. As the command speed increases to $[0.3, 0.6]$ $\mathrm{m/sec}$, the gait is changed to a *pace* and afterward seamlessly moves on to a *trot* for $[0.6, 0.9]$ $\mathrm{m/sec}$. We observe that the footfall frequency, as well as the duration ratio of swing to stance phases, increase with higher speed commands.

TABLE I: Parameters used for the experiments. Note that robot specific parameters such as joint torque limits are used as stated in the robots' datasheets.

| | |
|---|---|
| Gait planning time horizon $N_h$ | 60 (1 sec) |
| Gait planning time shift $t_{\text{shift}}$ | $0.5\,\mathrm{sec}$ |
| Base speed profile filter width | 5 (turning) / 7 (others) |
| Reference planner time horizon | $0.7\,\mathrm{sec}$ |
| Foot height threshold $\theta_{\text{height}}$ | $0.055\,\mathrm{m}$ |
| Foot velocity threshold $\theta_{\text{velocity}}$ | $0.8\,\mathrm{m/sec}$ |
| Cycle of motion matching $N$ | 30 |
| MPC Control Time step $\Delta t_{\text{MPC}}$ | $1/30\,\mathrm{sec}$ |
| MPC time horizon | $0.7\,\mathrm{sec}$ |
| MPC max./min. roll and pitch angle | $\pm 30\mathrm{deg}$ |
| MPC max./min. ground reaction force | $650\,\mathrm{N}$ / $0\,\mathrm{N}$ |
| WBC Control Time step $\Delta t_{\text{WBC}}$ | $1/120\,\mathrm{sec}$ |
| Simulation Time step | $1/480\,\mathrm{sec}$ |

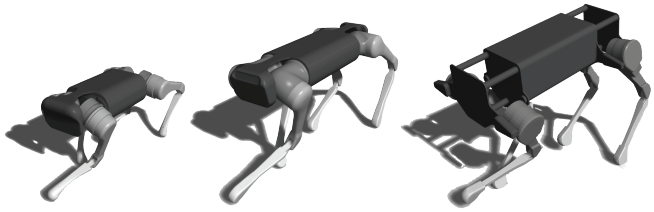

Fig. 4: Using our control pipeline, animal-like motions can be reproduced by various robot models with different scales and actuation power. We ran experiments with three robotic platforms from *Unitree*: *A1* [14] (**left**), *Aliengo* [12] (**middle**) and *Laikago* [15] (**right**).

## VI. Conclusion and Discussion

In this paper, we propose a control pipeline that enables legged robots to perform animal-like motions. It uses a simple data-driven approach to extract semantic information from animal motion data and embeds it in reference trajectories. A combination of MPC-WBC can track the reference trajectories effectively. As the semantic information is not constrained by morphology, our method can be generally applied to various quadrupedal robot models despite the discrepancy between the animal's and robots' morphology. We demonstrated the efficacy of our approach in simulation using several quadrupedal robot models with different dimensions and form factors. We observed that robots are able to reproduce key characteristics of animal-like movements such as subtle body movement variations, as well as nonperiodic gaits. As an added benefit of our approach, robots are able to naturally switch between different gaits based on the desired moving speed.

Our methodology, however, is not without limitations. First, since motion matching simply searches through the existing dataset without generating new movements, the resulting robot motions are limited by the richness of the motion capture dataset. We observed that the tracking performance of the target commands is highly dependent on the variety of sequences in the data. This problem can be addressed by either collecting more data or applying machine learning methods to extract underlying patterns in

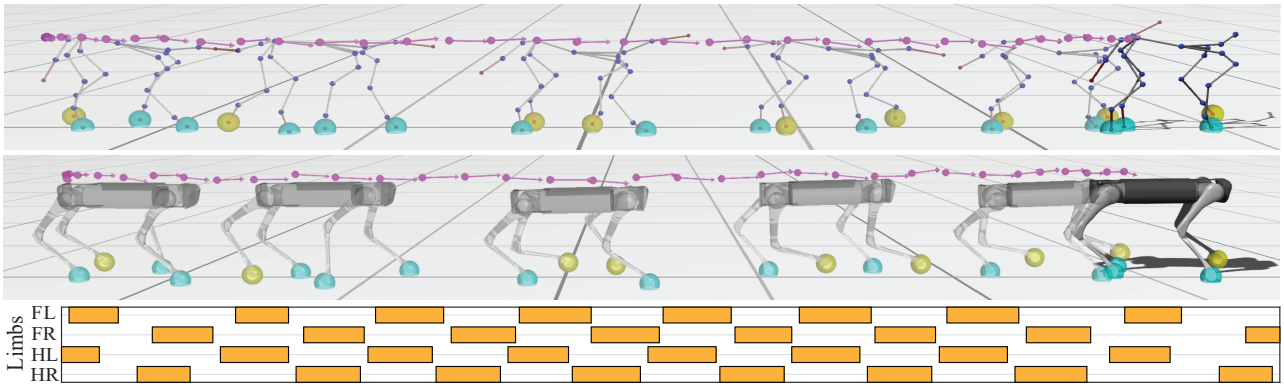


Fig. 5: A target motion sequence generated by motion matching (**top**) and reproduced by the Aliengo robot (**middle**) given a constant forward speed command of $0.8\mathrm{m/sec}$. Using our control pipeline, a robot can successfully execute a irregular gait sequence (**bottom**) with a varying base speed profile (in **purple**). Colored spheres around the feet visualize the phase of each limb (stance in **green**, and swing in **yellow**).

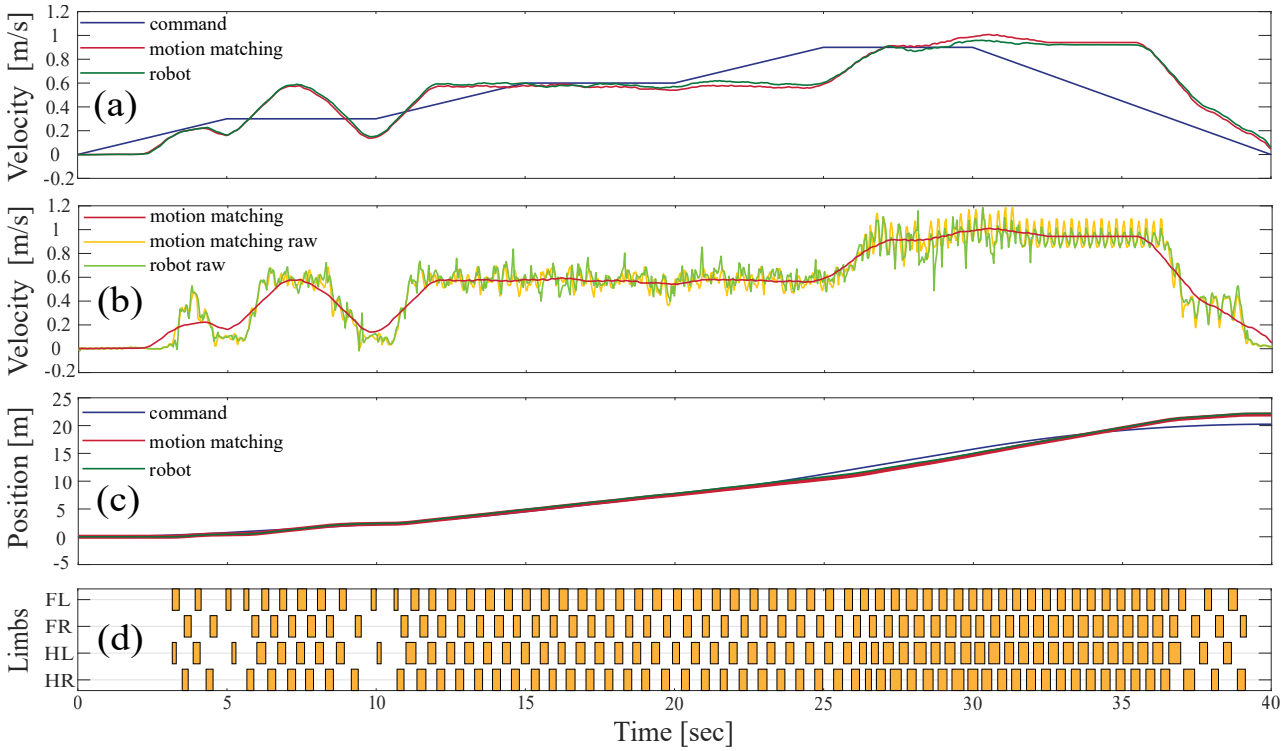


Fig. 6: Resulting motion behavior when applying a varying forward speed command (in **blue**) in the range of $[0, 0.9]$ $\mathrm{m/sec}$. The first two plots show the velocity profile with (a) and without (b) a moving average filter. The resulting position tracking is depicted in (c). (d) shows the executed footfall sequence.

the dataset [31]. Second, we noticed that *foot-skating* effects can occur during the motion matching phase, and these can result in infeasible footfall sequences that cannot be robustly handled by the locomotion controller. This is mainly due to the inertialization blending, which can propagate drift along the limbs. In practice, this is hardly problematic for slow and mid-range speed walking. However, for fast-moving or turning behaviors, the resulting foot-skate can be significant, which renders the foot velocity thresholding unsuitable for finding an appropriate footfall sequence. The graphics community has addressed this problem in the context of character animation [37], [39]. How these techniques can be adopted for our framework is part of future investigations.

Our immediate next step is to improve the control pipeline by addressing the aforementioned problems, and to deploy it on robots for hardware testing and validation. Our control pipeline is sufficiently efficient to be run in real-time, and the combination of MPC-WBC has already been demonstrated to run robustly on real-world robots [25]. Thus, we expect that our method can be successfully applied to hardware in the near future.

## REFERENCES


[1] H.-W. Park, P. M. Wensing, and S. Kim, "High-speed bounding with the mit cheetah 2: Control design and experiments," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, 2017.

[2] Q. Nguyen, M. J. Powell, B. Katz, J. Di Carlo, and S. Kim, "Optimized jumping on the mit cheetah 3 robot," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7448–7454.

[3] B. Katz, J. D. Carlo, and S. Kim, "Mini Cheetah: A Platform for Pushing the Limits of Dynamic Quadruped Control," in *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE, May 2019, pp. 6295–6301.

[4] E. Ackerman, "How Boston Dynamics Taught Its Robots to Dance - IEEE Spectrum," Jan. 2021.

[5] S. Varghese, "Welcome to the Uncanny Valley: how creepy robot dogs are on the rise - The New Statesman," Feb. 2018.

[6] Z. Li, C. Cummings, and K. Sreenath, "Animated cassie: A dynamic relatable robotic character," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3739–3746.

[7] S. Clavet, "Motion matching and the road to next-gen animation," in *Proc. of GDC*, 2016.

[8] D. Holden, O. Kanoun, M. Perepichka, and T. Popa, "Learned motion matching," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 53–1, 2020.

[9] K. Bergamin, S. Clavet, D. Holden, and J. R. Forbes, "Drecon: data-driven responsive control of physics-based characters," *ACM Transactions On Graphics (TOG)*, vol. 38, no. 6, pp. 1–11, 2019.

[10] D. Bollo, "High performance animation in Gears of War 4," in *ACM SIGGRAPH 2017 Talks*. Los Angeles California: ACM, Jul. 2017, pp. 1–2.

[11] M. H. Raibert, H. B. Brown Jr, and M. Chepponis, "Experiments in balance with a 3d one-legged hopping machine," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 75–92, 1984.

[12] Unitree Robotics, "Aliengo," https://www.unitree.com/products/aliengo.

[13] R. Smith *et al.*, "Open dynamics engine," 2005.

[14] Unitree Robotics, "A1," https://www.unitree.com/products/a1.

[15] ——, "Laikago pro," https://www.unitree.com/products/laikago.

[16] C. Gehring, S. Coros, M. Hutter, M. Bloesch, P. Fankhauser, M. A. Hoepflinger, and R. Siegwart, "Towards automatic discovery of agile gaits for quadrupedal robots," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4243–4248.

[17] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2665–2670.

[18] M. Hutter, H. Sommer, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart, "Quadrupedal locomotion using hierarchical operational space control," *The International Journal of Robotics Research*, vol. 33, no. 8, pp. 1047–1062, 2014.

[19] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 558–564.

[20] S. Fahmi, C. Mastalli, M. Focchi, and C. Semini, "Passive whole-body control for quadruped robots: Experimental validation over challenging terrain," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2553–2560, 2019.

[21] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[22] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT cheetah," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC: IEEE, Sep. 2017, pp. 4102–4109.

[23] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid: IEEE, Oct. 2018, pp. 1–9.

[24] Y. Ding, A. Pandala, and H.-W. Park, "Real-time Model Predictive Control for Versatile Dynamic Motions in Quadrupedal Robots," in *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE, May 2019, pp. 8484–8490.

[25] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control," *arXiv:1909.06586 [cs]*, Sep. 2019, arXiv: 1909.06586.

[26] N. Rathod, A. Bratta, M. Focchi, M. Zanon, O. Villarreal, C. Semini, and A. Bemporad, "Mobility-enhanced mpc for legged locomotion on rough terrain," *arXiv preprint arXiv:2105.05998*, 2021.

[27] 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 *Robotics: Science and Systems*, 06 2018.

[28] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019.

[29] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, 2020.

[30] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," in *Robotics: Science and Systems*, 07 2020.

[31] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–11, Aug. 2018.

[32] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "DeepMimic: example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–14, Aug. 2018.

[33] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine, "Sfv: Reinforcement learning of physical skills from videos," *ACM Trans. Graph.*, vol. 37, no. 6, Nov. 2018.

[34] N. Chentanez, M. Müller, M. Macklin, V. Makoviychuk, and S. Jeschke, "Physics-based motion capture imitation with deep reinforcement learning," in *Proceedings of the 11th annual international conference on motion, interaction, and games*, 2018, pp. 1–10.

[35] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, Jul. 2018.

[36] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[37] A. Treuille, Y. Lee, and Z. Popović, "Near-optimal character animation with continuous control," in *ACM SIGGRAPH 2007 papers*, 2007, pp. 7–es.

[38] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. van de Panne, "Locomotion skills for simulated quadrupeds," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 59:1–59:12, Jul. 2011.

[39] L. Kovar, J. Schreiner, and M. Gleicher, "Footskate cleanup for motion capture editing," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2002, pp. 97–104.