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

Dongho Kang, Simon Zimmermann, Stelian Coros

***Abstract*— Ongoing efforts in legged locomotion research have led to the development of mobile robots that can perform highly dynamic and agile locomotion maneuvers. However, despite their remarkable motor skills, legged robots' motions can not generally be considered *organic* or *life-like*. With the persistence of predesigned, repetitive motion patterns, people still perceive these robots as artificial machines. In this work, we propose a versatile and robust control pipeline that enables legged robots to exhibit more graceful and animal-like behaviors. To this end, we combine a state-of-the-art model-based controller with a simple data-driven approach that extracts semantic information of real-world animal motions. We demonstrate the efficacy of our method on a variety of quadrupedal robots in simulation. Our control pipeline is able to reproduce key characteristics of animal motions such as small body movements as well as non-periodic and skewed gait patterns. Most importantly, the robots naturally transition between different gaits based on the desired base velocity without any hard-coded rules.**

I. Introduction

Building robots that can reproduce the graceful and agile motions of different animals has been a long-sought vision of the robotics community. Inspired by nature, researchers have significantly expanded the skillset and enhance maneuverability of legged robots. This long endeavor is beginning to mature, and today's robots can perform highly dynamic and agile motions. State-of-the-art legged robots are capable of not only walking but can also perform highly energetic motions such as jumping [1], backflipping [2], and dancing [3]. Nevertheless, despite their remarkable capabilities, their exhibited behavior is arguably stiff and lifeless. Our goal is to make robots become more *alive* by conveying emotions through their behavior; in other words, we want to make robots "relatable" [4], [5]. From this point of view, we address the "naturality" of robots by adjusting their motions and behaviors to resemble the ones of real animals.

Quadrupedal robots follow symmetric, regular, and periodic footfall patterns in many of their standard walking modes while maintaining constant body height and velocity. These are usually predesigned or preprogrammed. In contrast, real animals adapt their walking patterns based on their target speed [6], [7]. Additionally, their bodies tend to shift or twist in sync with their footsteps slightly. Their gaits are non-periodic and skewed in a way that finding a governing rule and implementing it is hardly achievable.

In this work, we tackle this challenge by using motion capture data from real dogs moving at different speeds.

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

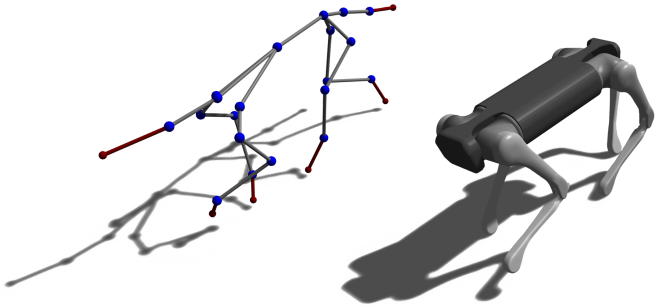

Fig. 1: A dog gait recorded by a motion capture system (**left**), reproduced by the quadrupedal robot *Aliengo* [8] (**right**).

This approach has two main advantages: First, it encourages natural transitions between different behaviors without time-consuming and restrictive manual generation of transition animations or state machines. Second, it preserves patterns of animal movements that people perceive as natural.

Using prerecorded motion sequences and mapping them to new behaviors is a problem that the computer graphics community has studied for character animation. There has been great success in generating realistic and natural-looking motions for human as well as animal characters [9], [10]. However, these techniques generally only attempt to create *plausible* animations. In contrast, robots are governed by real-world physics laws, which significantly restricts their motions. Furthermore, the motion patterns extracted from real animals' movements are typically non-symmetric, which poses an additional challenge for the locomotion controller to ensure safety and stability.

We present an approach that brings the best of both worlds: we combine character animation techniques with a state-of-the-art model-based controller for quadrupedal robots to generate more natural-looking motions. More specifically, we propose a control pipeline with four different stages. First, we apply common graphics tools, namely *motion matching* [9], [11] and *inertialization* [12] to plan the robot's gait based on a user-defined target base speed and real-world motion capture data. Then, we generate kinematic reference trajectories for the base and feet based on simple heuristics [13]. These are then optimized using a model-predictive controller (MPC) with a simplified model of the robot's dynamics that allows the robot to keep its balance while performing irregular footfall sequences. The final stage generates joint-level commands for the robot while maintaining a high update frequency to enable dynamic but robust maneuvers.

We demonstrate our results in physical simulation using

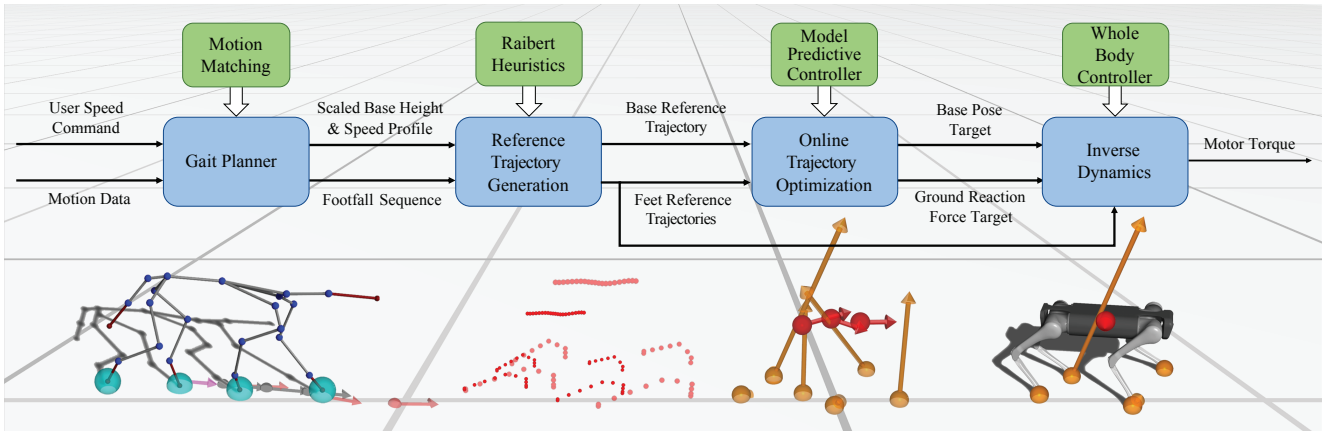

Fig. 2: Control pipeline overview. The individual stages (in **blue**) input and output different parameters (in **black**). The chosen methods are presented in **green**. The different stages are visualized in the row below.

the Open Dynamics Engine [14], and apply our technique to a variety of commercially available robot models [8], [15], [16]. We show that even with a simple tracking approach, the generated robot motions resemble real-world animal gaits.

II. Related Work

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

A. Quadrupedal Locomotion Control

To overcome the various challenges posed by legged locomotion, several control techniques have been vigorously studied and demonstrated to be successful on quadrupedal robots. One of these approaches is *model-based* control, which finds desired joint-level forces or accelerations by solving an inverse dynamics problem using an approximate dynamics model of the robot [17]–[19]. Recent efforts have extended this idea by integrating it into a model-predictive control (MPC) framework, where not only the next control step, but also an anticipation of the results of the robot's actions for a longer time horizon is considered. [20]–[24] show how MPC can be leveraged to perform highly dynamic and flexible quadrupedal maneuvers in various experiments. An alternative avenue is *learning-based* control, which creates controllers in a data-driven fashion instead. [25]–[27] propose learning pipelines to train a policy in simulated environments by applying numerous trial-and-error runs that can later be applied in the physical world. Meanwhile, [28] maps motion capture data of a real dog to a quadrupedal robot's morphology and train a policy to follow the joint trajectories. Since it is our goal to deploy our control pipeline on a variety of different quadrupedal robots, we choose a model-over a learning-based approach. It gives us more flexibility since the time-consuming training procedure does not need to be applied for each platform. Specifically, we integrate the combination of whole-body and model-predictive controller in the same manner as it is proposed in [22].

B. Character Animation

In our quest to generate more animal-like robot motions, we turn to the graphics community, which has done considerable work in this area. In this domain, it is common to use prerecorded motion capture or image data to generate motions that follow a user's high-level command. [11] 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. This method was extended later by applying deep learning techniques to improve data scalability [9]. Alternatively, [10] leverages deep neural networks to directly learn underlying patterns of a motion dataset, which can then be used to generate new motion sequences for a quadrupedal character. These approaches generate motions without considering physical laws thus require additional steps for transfer to real systems. This gap between the kinematic and the dynamic world has been addressed by *motion imitation* techniques as presented by [29], [30]. They learn policies that track target motions in a simulated physical environment. Consequently, the resulting motions are not only physically plausible but also have the potential to be transferred to real systems. [28] further extended this technique and applied it on a real robotic system as also mentioned in section II-A.

In our control pipeline, we leverage the motion matching technique [11] to generate target motions for our robots. The fundamental distinction of our work and motion imitation is that we extract semantic information from the motion database that can directly be used for trajectory planning. These semantics encode key characteristics of animal motions, and are used to create reference trajectories that our model-based controller can track. While motion imitation requires training a new controller for each robot and each behavior, our tracking approach can be directly applied to different robotic platforms, and seamlessly switch between different gaits based on the input base velocity.

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. An overview is given in Fig. 2. In the following subsections, we discuss the methodologies chosen for the last three stages. The gait planning stage 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 as well as its feet. The base trajectory, consisting of six degrees of freedom (three translational and three rotational) per time step, 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* [13]. 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_{stance}, assuming the robot is moving with constant velocity $\mathbf{v}_{\text{cur}}$. This results in identical landing and leaving angles for the individual limbs. 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_{raibert} that scales the impact of the velocity error. In addition, a centrifugal term is added as proposed in [22] to generate smoother trajectories under angular velocity command $\boldsymbol{\omega}_{\text{cmd}}$. In summary, the foot step location $\mathbf{r}_i$ for foot i is chosen as

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

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 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 tracking performance of the base trajectory and cause the robot to loose its balance and fall down 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. As proposed by [21], [22], [24], we apply a model-predictive control (MPC) scheme to achieve this. This technique holds the advantage that it considers a dynamics model of the robot instead of relying on kinematic information only. Furthermore, instead of tracking the target trajectory for only one control step, it predicts the robot's behavior for a long time horizon into the future, leading 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 therefore choose the MPC approach as presented in [21], [22] 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. Then, it finds an optimal base trajectory $\mathbf{x}$ as well as the ground reaction forces of the feet $\mathbf{f}$ for a given discrete time horizon m. The main advantage of this approach is that it can be formulated as a convex quadratic program (QP), the global optimal solution can be found efficiently. Note that this MPC formulation does not solve for feet trajectories, since optimization over feet placement in combination with ground reaction forces results in a non-convex problem [20], [31], which slows down the computation significantly and tends to get stuck in an undesirable local minimum.

Following the notation of [22], 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 are additionally limited by (2d). We ensure that the assumption of small roll ϕ and pitch θ angles is not violated by including the constraint (2e). We refer the reader to [21] and [22] for a more detailed derivation of the model and the problem formulation.

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 time step Δt_{MPC}. The resulting base trajectory $\mathbf{x}^{\text{MPC}}$ and ground reaction forces $\mathbf{f}^{\text{MPC}}$ is used as a target 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-level 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 (commonly at over 200 Hz) to allow the robot to perform dynamic maneuvers. Despite this requirement, it needs to take into

account the whole-body dynamics of the robot to ensure high accuracy. One controller that meets these requirements is the *whole-body controller (WBC)* [17]–[19]. Similarly, as in the previous stage, the WBC is formulated as a convex quadratic program to allow short computation times. The most noticeable difference to (2) is the dynamics model (2b), which is replaced by the more accurate whole-body dynamics of the robot. Additionally, the inverse dynamics problem only has to be solved for one control time step Δt_{WBC}. The whole-body dynamics model is parameterized using generalized accelerations and generalized forces, which we denote in the following by $\ddot{\mathbf{q}}$ and $\boldsymbol{\tau}$, respectively. Then, the QP can be formulated as

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

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

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

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

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

The optimization variables include the generalized acceleration $\ddot{\mathbf{q}}$, the generalized forces $\boldsymbol{\tau}$ as well as the ground reaction forces $\mathbf{f}$. The equality constraint (3b) is the equations of motion of the whole-body 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 the one 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 Δt is the control time step of WBC, $\mathbf{p}^{\text{target}}$ and $\dot{\mathbf{p}}^{\text{target}}$ are pose and velocity target, 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 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 (Eq. 1) can be avoided. This approach is also used by [21], [22] and has been found to work well in practice. 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.

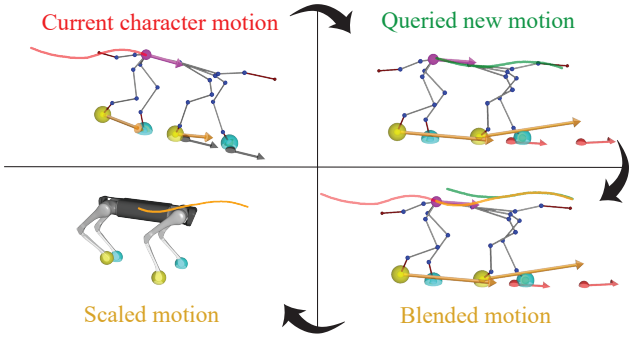

Fig. 3: Gait planning consists of several steps. Using the current character motion and the user input (**top left**), a new motion sequence is queried from the database (**top right**). To ensure a smooth transition, inertialization blending is applied (**bottom right**). Afterwards, we extract footfall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

IV. Gait Planning using Motion Matching

In this section, we present our gait planning strategy based on a simple data-driven method called *motion matching* [9], [11]. We integrate it into our legged robot control pipeline as the key component for generating organic motions for quadrupedal robots. As depicted in Fig. 2, the planner takes the user's target speed command (forward, sideways, and turning) and motion data of real animals as inputs and outputs the semantic information that encodes the key characteristics of the animal's motion. Precisely, we extract the footfall sequence, the base height, and the base speed profile. The base height and speed profile entail the animals' inconsistent moving speed as well as light up-and-down and left-and-right body offsets during a gait cycle. We observe that the timing of stepping and duration of swings of animal gaits is typically non-periodic, highly skewed, and involves natural transitions between different gait patterns. Thus, the animal's gait pattern can effectively be encoded in the footfall timings and the duration of contacts. 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.

A. Motion Matching

The main idea of motion matching is the following: given the current state of some predefined *motion features* and the user's target input, find the best matching motion sequence in the dataset. To do so, the entire dataset is searched in order to compare the motion features. We will now formulate this procedure for our specific case. Imagine we have a raw dataset consisting of n_c individual motion clips. We use it 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 jth 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 only contains pose data, we compute the generalized velocities $\dot{\mathbf{q}}$ using finite-difference.

From $\mathbf{y}_j$, we extract the corresponding *motion feature vector* $\mathbf{z}_j$ as described in [9] but modify the definition for quadrupedal characters as

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

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 feet positions and velocity, $\dot{\mathbf{p}} \in \mathbb{R}^3$ is the base velocity, and $\boldsymbol{\phi} \in \{0, 1\}^4$ is the contact states of all four feet, where 0 and 1 indicates 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, 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. It is used to retrieve the best matching feature vector $\mathbf{z}^\star$ in Z in terms of minimizing the euclidean distance with $\hat{\mathbf{z}}$. Based on $\mathbf{z}^\star$, the corresponding character state $\mathbf{y}^\star$ and its N_h subsequent states can be retrieved from Y.

B. Inertialization Blending

The new motion sequence found in 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* [12]. 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 [32]. 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 the quadrupedal character, we need an additional step to transfer this motion to the robot. This is because robots typically have a different morphology and actuation power than animals. Instead of applying a motion retargeting technique, we extract semantics of the character motions, namely the base speed and height profile, as well as the foot fall sequence. Consequently, the rest of the control pipeline can leverage this information instead of tracking joint trajectories or foothold of the character. This choice lets us propose a simple scaling scheme on speed and height profile to allow a robot with different dimensions and actuation power to use this semantic information. 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 for forward and sideways speed and a median filter for turning speed. We observe that this simplifies the tracking of the later-on 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 character's feet height and velocity. More formally, we consider a foot to be in swing mode if

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

where the specified thresholds for height and velocity θ_{height} and θ_{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 [33], 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_{shift} (see Table I). Specifically, we generate the gait plan for time $t + t_{\text{shift}}$ at control time t. It ensures that our controller does not abruptly switch between the currently executed and the 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 [14] and three different robotic platforms from *Unitree Robotics*: *A1* [15], *Aliengo* [8] and *Laikago* [16] 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. Furthermore, the dog motion capture data from [10] was used. In the following, we will present our findings using some representative motion sequences. For a more detailed visualization of these and additional results, we refer the reader to the accompanying video.

The setup for all conducted experiments was the same: The user can first select the preferred robot model and then introduce real-time target commands for the robot to follow using a simple GUI. These target commands are *forward*, *sideways* and *turning* velocity, expressed in the robot's base coordinate frame. The overall goal is that the robot adapts the organic and natural-looking motions of the real dog while following the target commands as closely as possible.

Fig. 5 shows the resulting motion of the Aliengo robot using a constant forward speed command. The robot started from a standing configuration and was then given a target forward speed of $0.8\,\mathrm{m/sec}$. Fig. 5 shows both the motion sequence of the motion capture dog and the robot. By observing the body trajectories of both creatures (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 as well, 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 non-periodic and involves a smooth and natural transition from a *walk* to a *pace* gait [34]. Both these properties stand in contrast to choosing a fixed, symmetric walking pattern. The visual differences between these behaviors are highlighted in the 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 portrays 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. 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 organic motions. The curve in light green shows that the robot can replicate this behavior well. 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. 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 afterwards 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_{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 θ_{height}	$0.055\,\mathrm{m}$
Foot velocity threshold θ_{velocity}	$0.8\,\mathrm{m/sec}$
Cycle of motion matching N	30
MPC Control Time step Δt_{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 Δt_{WBC}	$1/120\,\mathrm{sec}$
Simulation Time step	$1/480\,\mathrm{sec}$

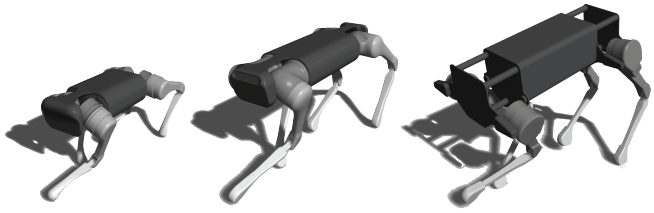

Fig. 4: Using our control pipeline, animal-like motions can be reproduced by various robot models with different scales and actuation power. We ran experiments with three robotic platforms from *Unitree*: *A1* [15] (**left**), *Aliengo* [8] (**middle**) and *Laikago* [16] (**right**).

VI. Discussion And Future Work

In this paper, we propose a control pipeline that enables legged robots to perform organic, animal-like motions. It uses a simple data-driven approach to extract semantic information from real-world animal motion data and embeds it in kinematic reference trajectories for the robot. A combination of MPC and WBC can track the reference trajectories effectively. We demonstrated the efficacy of our approach

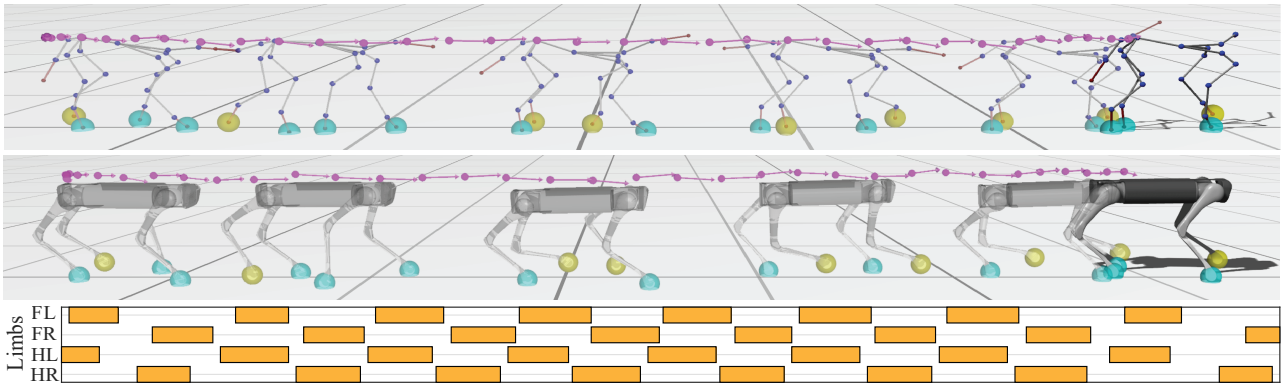

Fig. 5: An animal-like target motion sequence generated by motion matching (**top**) and reproduced by the Aliengo robot (**middle**) given a constant forward speed command of $0.8\mathrm{m/sec}$. Using our control pipeline, the robot can successfully execute a irregular gait sequence (**bottom**) with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

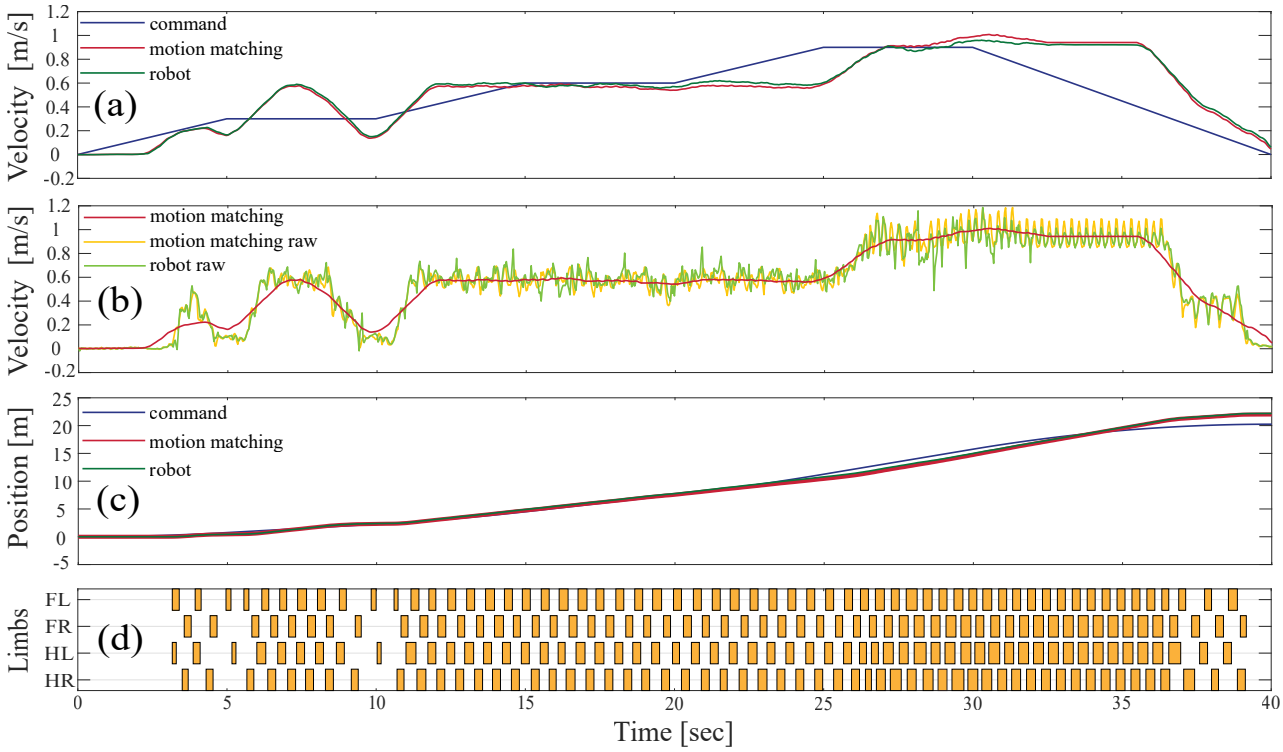

Fig. 6: Resulting motion behavior when applying a varying forward speed command (in blue) in the range of $[0, 0.9]$ $\mathrm{m/sec}$. The first two plots show the velocity profile with (a) and without (b) moving average filter. The resulting position tracking is depicted in (c). (d) shows the executed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

in simulation using several quadrupedal robot models with different dimensions and form factors. We observed that the robots are able to reproduce key characteristics of animal-like movements such as small body movements as well as non-periodic and irregular foot steps. Most importantly, the robots transition naturally between different gates based on the desired moving speed of the base without any hard-coded rules.

Nevertheless, our methodology bears several limitations. First, since motion matching simply searches through the existing dataset without generating new movements, the resulting robot motions are limited by the quality of the 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 by applying machine learning methods to extract underlying patterns of the dataset [10]. Second, we noticed that foot-skating effects can occur during the motion matching phase, which can result in ”unreasonable” footfall sequences that cannot be 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, the resulting foot-skate can be significant, which renders the foot velocity thresholding to be unsuitable for finding an appropriate footfall sequence. The graphics community has addressed this problem in the context of character animation [33], [35]. How these techniques can be adopted for our framework is part of future investigations. Finally, we found that our current implementation of the MPC-WBC locomotion controller is not yet powerful enough to perform more dynamic animal maneuvers such as hoping, quick turns, or galloping. Therefore, a deeper investigation into these control frameworks is necessary to increase versatility and robustness further.

Our immediate next step is to improve the control pipeline by addressing the aforementioned problems and deploy it to real robot hardware. Our control pipeline is computationally efficient to run in real-time, and the combination of MPC and WBC has already been demonstrated to run robustly on real-world robots [22]. Thus, we expect that our method can be successfully applied to hardware in the near future.

REFERENCES

[1] 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.

[2] 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.

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

[4] T. Shibata and K. Tanie, "Physical and affective interaction between human and mental commit robot," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 3. IEEE, 2001, pp. 2572–2577.

[5] Z. Li, C. Cummings, and K. Sreenath, "Animated Cassie: A Dynamic Relatable Robotic Character," *arXiv:2009.02846 [cs]*, Sep. 2020, arXiv: 2009.02846.

[6] D. F. Hoyt and C. R. Taylor, "Gait and the energetics of locomotion in horses," *Nature*, vol. 292, no. 5820, pp. 239–240, 1981.

[7] D. Owaki and A. Ishiguro, "A quadruped robot exhibiting spontaneous gait transitions from walking to trotting to galloping," *Scientific reports*, vol. 7, no. 1, pp. 1–10, 2017.

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

[9] 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.

[10] 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.

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

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

[13] 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.

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

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

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

[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, Jul. 2014, publisher: SAGE Publications Ltd STM.

[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] 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.

[21] 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.

[22] 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.

[23] G. Bledt and S. Kim, "Implementing Regularized Predictive Control for Simultaneous Real-Time Footstep and Ground Reaction Force Optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Macau, China: IEEE, Nov. 2019, pp. 6316–6323.

[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] 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," *arXiv preprint arXiv:1804.10332*, 2018.

[26] 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.

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

[28] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning Agile Robotic Locomotion Skills by Imitating Animals," *arXiv:2004.00784 [cs]*, Apr. 2020, arXiv: 2004.00784.

[29] 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.

[30] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine, "SFV: Reinforcement Learning of Physical Skills from Videos," *arXiv:1810.03599 [cs]*, Oct. 2018, arXiv: 1810.03599.

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.