Supplementary Material for Adaptive Tracking of a Single-Rigid-Body Character in Various Environments

1 DETAILS OF LQR FILTERING FOR SWING FOOT STATE

The global desired foot landing position $\check{\mathbf{f}}_t^j$ is calculated from $\mathbf{a}_s^j$, the desired foot landing position in the forward-facing SRB frame, as follows:

$$\check{\mathbf{f}}_t^j = \text{projY}\left(\mathbf{p}_t^{\{g\}} + \mathbf{R}_t^{y\{g\}}\left(\left(\mathbf{a}_s^j\right)_t + \mathbf{o}_t^j\right)\right), \tag{1}$$

where $\mathbf{R}_t^{y\{g\}}$ refers to the orientation of the center of mass consisting only of rotation about the global vertical axis. $\mathbf{o}_t$ is the foot offset vector that aligns the desired landing position $\check{\mathbf{f}}_t^j$ with the reference SRB motion's landing position when action $\mathbf{a}_s^j = \mathbf{0}$ (refer to Section 1.1 for the calculation of $\mathbf{o}_t$). Function projY$(\cdot)$ projects the 3D global position onto the horizontal ground plane. The global desired foot landing orientation $\check{\mathbf{F}}_t^j$ is directly obtained from the reference SRB motion.

Then, the foot state of the SRB character in a swing state is calculated as follows:

$$\mathbf{f}_t^j, \dot{\mathbf{f}}_t^j = \text{LQRfilter}\left(\mathbf{f}_{t-1}^j, \dot{\mathbf{f}}_{t-1}^j, \check{\mathbf{f}}_t^j, \alpha\right), \tag{2}$$

$$\mathbf{F}_t^j, \dot{\mathbf{F}}_t^j = \text{LQRfilter}\left(\mathbf{F}_{t-1}^j, \dot{\mathbf{F}}_{t-1}^j, \check{\mathbf{F}}_t^j, \alpha\right). \tag{3}$$

Here, $\mathbf{f}_t^j$ and $\mathbf{F}_t^j$ are the continuously changing global position and orientation of each foot, which are LQR-filtered values of $\check{\mathbf{f}}_t^j$ and $\check{\mathbf{F}}_t^j$ (refer to Section 1.2 for LQRFilter), and $\dot{\mathbf{f}}_t^j$ and $\dot{\mathbf{F}}_t^j$ denote their time derivatives. The parameter α, which adjusts the strength of the LQR filtering, is experimentally set to 8 in our experiments.

1.1 Foot offset vector $\mathbf{o}_t$

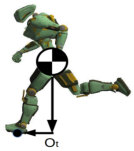


Fig. 1. Foot offset vector $\mathbf{o}_t$.

The foot offset vector $\mathbf{o}_t$ used in Equation 1 adjusts the desired landing position of the j^{th} foot at touch down to the landing position of the reference SRB motion when action $\mathbf{a}_s^j = 0$ (Figure 1). This is calculated as follows:

$$\mathbf{o}_t^j = \left(\hat{\mathbf{R}}_{\psi(t)}^{y\{g\}}\right)^{-1} \cdot \left(\hat{\mathbf{f}}_{\psi(t)}^j - \text{projY}\left(\hat{\mathbf{p}}_{\psi(t)}^{\{g\}}\right)\right), \tag{4}$$

where $\hat{\mathbf{R}}_{\psi(t)}^{y\{g\}}$ is a rotational matrix that only contains the rotational components of the reference SRB motion's center of mass with respect to the global vertical axis, $\hat{\mathbf{f}}_{\psi(t)}^j$ is the global position of the j^{th} foot of the reference SRB motion, and $\hat{\mathbf{p}}_{\psi(t)}^{\{g\}}$ is the global position of the reference SRB motion's center of mass. All values are obtained for time $\psi(t)$ of the reference SRB motion that corresponds to the simulation time t. For reference, most motions can be learned when no offset is given ($\mathbf{o}_t = 0$), but the learning speed is faster when $\mathbf{o}_t$ calculated from Equation 4 is given. Additionally, some motions may not be well learned when no offset is given.

Author's address:

1.2 LQRFilter

Our LQRFilter dynamically controls the mass particles, which represent the continuously changing positions and orientation of the feet, to follow the discontinuously changing desired landing positions and orientation.

Specifically, 1D mass particle dynamics simulation is used in our LQRFilter. Three 1D mass particles are used for each leg to express the foot position (2D) and orientation (1D). The foot orientation is converted by a single scalar rotation angle θ to be expressed as a single mass particle, and used as an input to LQRFilter, and the filtered output is converted back into a rotation matrix. Here, θ is expressed as an angle relative to the frontal direction of the SRB to avoid singularity.

Linear quadratic regulators are used to control the mass particles. The equation of motion for each particle can be expressed as the state space equation of a continuous-time linear system: $\dot{\mathbf{s}} = \mathbf{A}\mathbf{s} + \mathbf{B}\mathbf{u}$. For example, $\mathbf{s}$ is a 2D vector that stores the position and speed of a particle. The problem of physically controlling this particle using continuously changing external force $\mathbf{u}$ can be expressed as an optimal control problem using a linear quadratic regulator (LQR), and the control results in the particle moving along a smooth C^2 continuous curve. If the control objective function J for guiding the particle to the origin is defined as $J = \int_0^\infty \left(\mathbf{s}^T\mathbf{Q}\mathbf{s} + \mathbf{u}^T\mathbf{R}\mathbf{u}\right) dt$, the linear feedback matrix $\mathbf{K}$ that minimizes J has an analytic solution [Dorato et al. 1994]. By applying the linear feedback law $\mathbf{u} = -\mathbf{K}(\mathbf{s} - \mathbf{s}_d)$ to calculate the external force to guide the particle to the desired position, it is possible to calculate the continuously changing foot position and orientation (represented as particle states $\{\mathbf{s}\}$) of the SRB character that follows the discontinuous desired foot landing position and orientation (represented as desired particle states $\{\mathbf{s}_d\}$).

For instance, when considering a 1D particle representing the x-position of a foot, the desired particle state $\mathbf{s}_d$ is a two-dimensional vector, where the x-coordinate of the desired foot landing position is its position and its speed is set to 0. By changing the positional component $Q_0 = 10^\alpha$ of the diagonal weighting matrix $\mathbf{Q}$, the speed at which how fast the particle moves towards its destination can be controlled. By using a large value for α, the mass particle closely follows the desired foot landing position with minimal filtering.

2 MOTION PHASE ADJUSTMENT DETAILS

Stride adjustment. The phase change rate is adjusted relative to the locomotion speed to prevent excessively lengthy strides. Let Δv be the difference between the velocities of the reference and simulated centers of mass of the SRB character ($\Delta v = \|\Delta\mathbf{p}_t - \Delta\hat{\mathbf{p}}\|$).

Where the average moving speed of the reference motion is $\bar{v}$ and the reference phase change rate is $d\hat{\psi}/dt$, the phase change rate in the SRB simulation is increases as Δv increases:

$$\frac{d\psi}{dt} \leftarrow \frac{d\hat{\psi}}{dt}\left(1 + \beta \Delta v \bar{v}\right). \tag{5}$$

In general, because Δv converges to 0 during the optimization process, the phase change rate is usually close to the reference phase change rate with deviations in the phase change rate only when there are external forces or terrain. The motion style can be changed by adjusting the weight β, which is experimentally set to 0.4 in our experiments.

Contact timing adjustment. If an unexpectedly strong external force acts on the character, the character cannot touch down at the specified timings, failing to stably control the character. To overcome this issue, we present a method to control the motion phase $\psi(t)$ by detecting the timing of early or late touch downs. When the height y_t of the SRB character's center of mass is greater than the height $\hat{y}_{\psi(t)}$ of the reference SRB motion's center of mass beyond the given threshold (0.05 m) at the timing when touch down should have happened, it is considered that the contact between the foot and ground has not been made well yet, and is thus regarded as a late contact. In such instances, we cut the phase change rate $d\psi/dt$ in half to delay the contact timing. Conversely, where y_t was smaller than $\hat{y}_{\psi(t)}$ beyond the threshold (0.1 m) before the contact timing of the SRB character, this is regarded as early touch down. In such instances, we double $d\psi/dt$ to allow contact before the given timing.

An earlier study [Lee et al. 2010] also presented adjusting the phases of reference motions based on the simulated character's contact state. In their study, the phase of the reference motion is discontinuously adjusted upon touchdown and the contact time is strictly governed by the full-body simulator. In contrast, our simulated character follows the reference SRB motion, allowing for a more gradual adjustment of the motion phase, resulting in stable balancing of the character.

3 DELTA COMPUTATION DETAILS

Calculate COM frame delta. The average transformation differences between the baseline SRB motion's COM frame and the full-body reference motion's COM frame are calculated for each different motion phase in $0 \leq \psi(t) \leq 2\pi$. The average delta transformation is calculated as per $\psi(t)$ by Equation 6:

$$\Delta\bar{\mathbf{T}}_{\psi(t)} = \frac{\sum_{c_n} \log\left(\bar{\mathbf{T}}_t^{-1} \cdot \check{\mathbf{T}}_{\psi(t)}\right)}{c_n}, \tag{6}$$

where c_n refers to the number of cycles included in the baseline SRB motion, $\bar{\mathbf{T}}_t$ the baseline motion COM frame (i.e., SRB Frame) at time t, and $\check{\mathbf{T}}_{\psi(t)}$ the full-body reference motion's COM frame (i.e., reference SRB frame $\hat{\mathbf{T}}_{\psi(t)}$) at the corresponding time point. The calculated $\Delta\bar{\mathbf{T}}_{\psi(t)}$ is multiplied to the SRB frame of the SRB character in runtime simulation to calculate the target posture of MMIK, where the values stored in twist format are transformed into a rigid-body transformation by the exponential map.

Calculate contact points delta. The baseline SRB motion involves smooth changes in the foot's position and orientation, but the feet remain in contact with the ground since it doesn't include foot height. In order to convert this into a natural human foot movement, the average positions of the contact points of each foot of the full-body reference motion, expressed in the corresponding foot frame of the baseline SRB motion, are calculated for each different motion phase as follows:

$$\Delta\bar{\mathbf{c}}^{jk}_{\psi(t)} = \frac{\sum_{c_n} \mathrm{F}\left(\bar{\mathbf{F}}^j_t, \bar{\mathbf{f}}^j_t\right)^{-1} \cdot \check{\mathbf{c}}^{jk\{g\}}_{\psi(t)}}{c_n}, \tag{7}$$

where $\bar{\mathbf{f}}^j_t$ and $\bar{\mathbf{F}}^j_t$ refer to the global position and orientation of the j^{th} foot in time t for the baseline SRB motion, and $\check{\mathbf{c}}^{jk\{g\}}_{\psi(t)}$ refers to the global positions of the k^{th} contact points of the j^{th} foot of the corresponding full-body reference motion. The calculated $\Delta\bar{\mathbf{c}}^{jk}_{\psi(t)}$ is applied to the foot state of the SRB character in the runtime simulation to calculate the target contact positions of MMIK.

Calculate centroidal velocity delta. The average differences between the linear and angular velocities of the baseline SRB motion and the centroidal linear and angular velocities of the full-body reference motion are calculated for each different motion phase as follows:

$$\Delta\bar{\mathbf{v}}_{\psi(t)} = \frac{\sum_{c_n} \mathrm{Ad}_{\left(\mathbf{R}^{y\{g\}}_t\right)^\top}\left(\mathbf{I}_f^{-1}\mathbf{J}_m\dot{\check{\mathbf{x}}}_{\psi(t)} - \bar{\mathbf{v}}_t\right)}{c_n}. \tag{8}$$

Here, $\mathbf{I}_f$ refers to the complex rigid body inertia matrix of the full-body character, while momentum Jacobian $\mathbf{J}_m$ relates the full-body generalized velocity to the generalized centroidal momentum, $\bar{\mathbf{v}}_t$ refers to the global linear and angular velocities of the baseline SRB motion at time t, and $\dot{\check{\mathbf{x}}}_{\psi(t)}$ the generalized velocity of the full-body reference motion at the corresponding time. Thus, $\mathbf{I}_f^{-1}\mathbf{J}_m\dot{\check{\mathbf{x}}}_{\psi(t)}$ refers to the full-body reference motion's centroidal velocity. As $\mathbf{I}_f$, $\mathbf{J}_m$, and $\bar{\mathbf{v}}_t$ are calculated in the global coordinate system, to convert $\Delta\bar{\mathbf{v}}_{\psi(t)}$ in reference to the forward-facing SRB frame, the Adjoint map $\mathrm{Ad}_{\left(\mathbf{R}^{y\{g\}}_t\right)^\top}$ for the inverse of the COM orientation only about the global vertical axis $\mathbf{R}^{y\{g\}}_t$ and the zero translation vector $\mathbf{0}$ is used.

4 MMIK DETAILS

MMIK minimizes the following cost function:

$$\begin{aligned} E_{IK}(\mathbf{x}) = &\sum_j \sum_k \|\mathbf{C}^{jk}(\mathbf{x}) - \mathrm{F}\left(\mathbf{F}^j_t, \mathbf{f}^j_t\right) \cdot \Delta\bar{\mathbf{c}}^{jk}_{\psi(t)}\|^2 \\ &+ w_g\|\mathbf{I}_f^{-1}\mathbf{J}_m(\mathbf{x} - \bar{\mathbf{x}})\|^2 \\ &+ w_m\|\mathbf{I}_f^{-1}\mathbf{J}_m\dot{\mathbf{x}} - \left(\mathbf{v}_t + \mathrm{Ad}_{\mathbf{R}^{y\{g\}}_t}\Delta\bar{\mathbf{v}}_{\psi(t)}\right)\|^2 \\ &+ w_p\|v_p - \mathbf{N}_p\mathbf{J}_p\dot{\mathbf{x}}\|^2_{\oplus} + w_v\|\dot{\mathbf{x}} - \dot{\bar{\mathbf{x}}}\|^2 + w_r\|\mathbf{x} - \bar{\mathbf{x}}\|^2, \end{aligned} \tag{9}$$

where $\mathbf{x}$ is the full-body pose, which is the optimization variable, and $\dot{\mathbf{x}}$ is the time derivative of the full-body pose calculated via backward differentiation. The full-body desired pose $\bar{\mathbf{x}}$ is obtained by applying rigid transformation to the full-body reference pose to align its COM frame $\check{\mathbf{T}}_{\psi(t)}$ with the COM frame $\bar{\mathbf{T}}_t \cdot \exp(\Delta\bar{\mathbf{T}}_{\psi(t)})$ of the simulated SRB character corrected with the COM frame delta

calculated from Equation 6. Its time-derivative $\dot{\mathbf{x}}$ is also calculated by backward differentiation.

The first term measures the difference between the position of the k^{th} contact point of the j^{th} foot of the full-body pose $\mathbf{x}$ ($\mathbf{C}^{jk}(\mathbf{x})$) and that of the simulated SRB character calculated by transforming the swing foot delta $\Delta\bar{\mathbf{c}}^{jk}_{\psi(t)}$ by the rigid transformation constructed form the position $\mathbf{f}^j_t$ and orientation $\mathbf{F}^j_t$ of the simulated SRB character's j^{th} foot.

The second term, based on momentum-based geometric mapping [Kwon and Hodgins 2017], aims to find $\mathbf{x}$ for which a change from $\mathbf{x}$ to $\bar{\mathbf{x}}$ does not cause any centroidal velocity. $\mathbf{I}_f$ is the composite rigid body inertia matrix of the full-body character in the pose $\mathbf{x}$, and the momentum Jacobian $\mathbf{J}_m$ relates the full-body's generalized velocity to the generalized centroidal momentum.

The third term measures the difference between the global linear and angular velocity $\mathbf{v}_t + \mathrm{Ad}_{\mathbf{R}_t^{y\{g\}}} \Delta\bar{\mathbf{v}}_{\psi(t)}$ of the simulated SRB motion calibrated by the centroidal velocity delta $\Delta\bar{\mathbf{v}}_{\psi(t)}$ and the centroidal velocity $\mathbf{I}_f^{-1}\mathbf{J}_m\dot{\mathbf{x}}$ calculated from $\dot{\mathbf{x}}$. To convert $\Delta\bar{\mathbf{v}}_{\psi(t)}$ calculated in the forward-facing SRB frame to the global frame, Adjoint map $\mathrm{Ad}_{\mathbf{R}_t^{y\{g\}}}$ was used.

The fourth term is only used when external forces are applied on the full-body character, which creates more natural responses to such forces. This term constrains the point at which the force is applied to move faster than v_p in the direction $\mathbf{N}_p$ of the applied force. v_p is the heuristic speed threshold set in proportion to the magnitude of the external force. $\|\cdot\|_{\oplus}$ is assessed as 0 when the internal $\cdot$ has a negative value, and $\mathbf{J}_p$ is the Jacobian for the point of application of the external force. This term uses redundant DOFs to create a natural response to the external force. Without the term, a non-compliance response will be generated where the whole body will bend uniformly under the external force.

The last two terms are regularization terms that maintain the solution close to the desired full-body pose $\bar{\mathbf{x}}$ and its time-derivative $\dot{\bar{\mathbf{x}}}$.

5 IMPLEMENTATION AND TRAINING DETAILS

Implementation. The quadratic programming solver was implemented in a Python 3.10.6 environment, and policy and value function network were implemented and trained using Pytorch ver.1.12.1.

Policy network. Our policy network consists of 64 hidden units, configured into two fully connected layers and one output layer. All hidden units used tanh for their activation function. The value network is also configured in the same structure. While the policy network has as many linear output units as the number of action dimensions, the value network has a single linear unit. The policy was updated at every m = 1024 samples and mini-batch size n = 256 was used. A discount factor $\gamma = 0.995$ was applied.

Episode termination condition. To ensure an efficient learning journey, an episode should be immediately terminated when the simulation meets certain criteria and move onto the next episode. In this study, the episodes were set to terminate when the height of the center of mass was too high or low, making it difficult to properly track the reference motion, or when the vertical axis (y-axis) of the SRB frame has an incline of more than 70 degrees against the global y-axis. The valid height range for the center of mass was set to $\min(0.7 \cdot \hat{y}_{\psi(t)}, 0.2) < y_t < 2.0$, where y_t refers to the height of the SRB character's center of mass at time t and $\hat{y}_{\psi(t)}$ refers to that of the reference SRB motion. Also, episodes were set to terminate when they exceeded 3 seconds.

Foot touch down / off time in the blending experiments. The remaining time in $\hat{\mathbf{m}}_A$ and $\hat{\mathbf{m}}_B$ until each j^{th} foot touches down $t_A^{d,j}$, $t_B^{d,j}$ and until touch off $t_A^{o,j}$, $t_B^{o,j}$ are expressed as signed distances so that interpolation is possible when the contact states of Controller A and Controller B are different, and the contact state change (touch down or touch off) in the blended controller occurs at the point when the interpolated distance becomes 0.

6 ADDITIONAL EXPERIMENTAL RESULTS

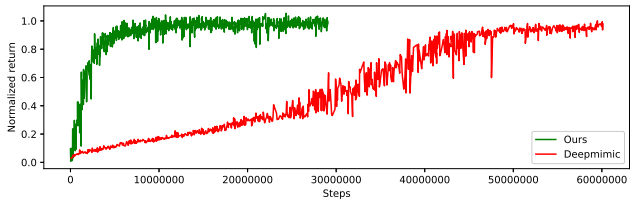
6.1 Sample efficiency

Our approach has advantages in terms of adaptability and sample efficiency because it is based on simplified physics. However, since it does not rely on full-body dynamics, it cannot fully represent general contact-rich scenarios. In contrast, DeepMimic [Peng et al. 2018], being based on full-body dynamics, can accurately represent various motions in general physically-interacting situations, but it has lower adaptability due to full-body tracking and requires a large number of samples for learning due to the high dimensionality. Given these fundamental differences in their characteristics, it is not appropriate to compare the two methods directly. However, for reference purposes, we would like to discuss their sample efficiency using learning curves.

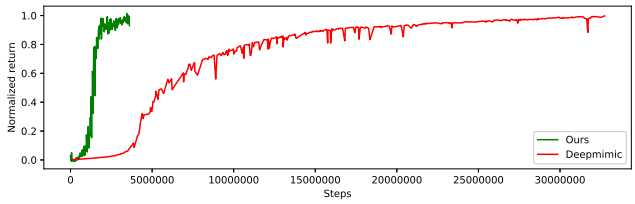
Figure 2 shows the learning curves of our method and DeepMimic for four reference motions. It can be seen that our method can learn the control policy with a smaller number of samples than DeepMimic. Specifically, ours requires about 8% (*Sprint*) to 38% (*Walk*) samples for the policy to converge. The number of samples required for convergence with DeepMimic varies greatly depending on the dynamics of the reference motion (*Walk*: about 7.4×10^6, *Sprint*: about 3×10^7, *Sprint jumps*: about 5×10^7), while our method has a relatively small difference in the number of samples required for different reference motions (*Walk*: about 2.8×10^6, *Sprint*: about 2.5×10^6, *Sprint jumps*: about 7×10^6). We attribute this sample efficiency to the fact that our policy is not trained to output detailed full-body actions based on the detailed state of the full-body, but rather to output the overall desired velocity and contact position of a single rigid body based on the overall state of the single body. Our method is faster in terms of wall-clock time as well, as it can collect more samples during the same period (ours: about 2000 steps per second, DeepMimic: about 1800 steps per second).

6.2 Runtime computational statistics

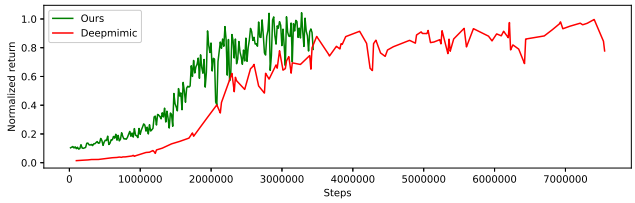
Table 1 presents the runtime computational statistics for our method and [Kwon et al. 2020]. Our controllers can generate a full-body motion approximately four times faster than real time, and a large part of the computation time is spent solving MMIK. Note that learning is performed much faster because there is no need to solve MMIK during learning. For a rough comparison, we included the runtime statistics from [Kwon et al. 2020], indicating the time required to generate motions similar to ours as closely as possible. In their work,



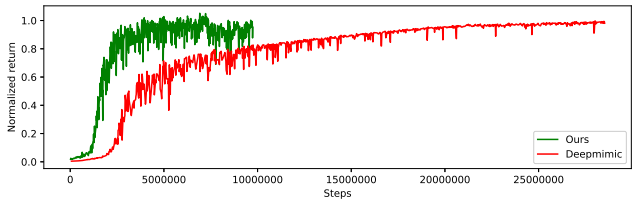
(a) *Sprint jumps*



(b) *Sprint*



(c) *Walk*



(d) *Run*

Fig. 2. Learning curves comparing our method and our implementation of DeepMimic.

real-time generation of some motions is possible, but generating faster or shorter stance motions through trajectory optimization requires more time due to increased complexity and longer unactuated phases.

6.3 MMIK with and without precomputed delta

To validate the impact of the COM frame delta $\Delta\bar{\mathbf{T}}_{\psi(t)}$ and the centroidal velocity delta $\Delta\bar{\mathbf{v}}_{\psi(t_p)}$ (described in Section 3) on the full-body motion quality generated by MMIK, we have compared the simulation results where such terms were turned On/Off. The upper body movement increased to the point of looking unsteady when the COM frame delta were switched off, while the upper body and the head shook unnaturally when the centroidal velocity delta was turned off. These differences can be best observed in the accompanying video.

Table 1. Statistics for generating locomotion on a flat ground. In the table, average computation time for generating one second of final motion is measured. Total time includes all the necessary computation time except the rendering time.

average computation time for 1s full-body motion				
		Walk	Sprint	Run180
Ours	policy output	0.007s	0.007s	0.007s
	QP simulation	0.061s	0.034s	0.042s
	MMIK(with Δ)	0.23s	0.18s	0.19s
	total	0.29s	0.22s	0.24s
		Walk	Fast run	Run
[Kwon et al. 2020]	motion sketch	0.07s	0.18s	0.10s
	CDM planning	0.16s	1.08s	0.13s
	MMIK	0.23s	0.23s	0.23s
	total	0.46s	1.49s	0.46s

REFERENCES

Peter Dorato, Vito Cerone, and Chaouki Abdallah. 1994. *Linear-Quadratic Control: An Introduction.* Simon & Schuster.

Taesoo Kwon and Jessica K Hodgins. 2017. Momentum-mapped inverted pendulum models for controlling dynamic human motions. *ACM Transactions on Graphics (TOG)* 36, 1 (2017), 1–14.

Taesoo Kwon, Yoonsang Lee, and Michiel Van De Panne. 2020. Fast and flexible multilegged locomotion using learned centroidal dynamics. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 46–1.

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-driven biped control. *ACM Trans. Graph.* 29, 4 (2010), 1–8.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.