Supplementary Material for MuscleVAE: Model-Based Controllers of Muscle-Actuated Characters

Yusen Feng
ysfeng@stu.pku.edu.cn
School of CS, Peking University
Beijing, China

Xiyan Xu
xiyan_xu@pku.edu.cn
Peking University
Beijing, China

Libin Liu*
libin.liu@pku.edu.cn
Peking University
Beijing, China
National Key Lab of General AI
Beijing, China

A MUSCLE MODELING

A.1 Muscle Routing

We approximated the muscles of the Hill-type model as polylines, with the inflection points of these polylines serving as the muscles' anchor points. The positioning of the muscle anchors can be characterized by LBS (Linear Blend Skinning), as mentioned in the main article. These anchor points play a crucial role in determining the muscle's route and also provide the location where the mid-muscle contraction force can be transferred to the bones. Specifically, the position of the anchor point is

$$\boldsymbol{p} = \sum w_i T_i \boldsymbol{x}_i', \tag{18}$$

where $\boldsymbol{p}$ denotes the position of the muscle anchor point, $\boldsymbol{x}_i'$ represents the relative position of the anchor point to the i-th bone. The local coordinate system of the i-th bone can be represented as a translation-rotation matrix, denoted as T_i, in the global coordinate system. The variable w_i is the weight of the anchor point with respect to the i-th bone, which is computed based on the distance between the anchor point and the bone, as suggested by Lee et al. [2019]. Leveraging the anchor positions, we can calculate the muscle length as

$$l_{\mathrm{M}} = \sum_{k=1}^{n-1} \|\boldsymbol{s}_k\|, \ \boldsymbol{s}_k = \boldsymbol{p}_{k+1} - \boldsymbol{p}_k, \tag{19}$$

where n is the number of anchor points of the muscle.

We utilize a muscle dynamics model to compute the amplitude of a muscle's force, denoted by f_m, as detailed in the next section. The forces applied at each anchor point are then computed as

$$\boldsymbol{f}_k^- = f_{\mathrm{m}} \frac{\boldsymbol{p}_{k-1} - \boldsymbol{p}_k}{\|\boldsymbol{p}_{k-1} - \boldsymbol{p}_k\|}, \quad \boldsymbol{f}_k^+ = f_{\mathrm{m}} \frac{\boldsymbol{p}_{k+1} - \boldsymbol{p}_k}{\|\boldsymbol{p}_{k+1} - \boldsymbol{p}_k\|}. \tag{20}$$

In this expression, $\boldsymbol{f}_k^-$ and $\boldsymbol{f}_k^+$ represent the forces exerted along the two polylines that join at the k-th anchor point of the muscle, respectively. Notably, for endpoint anchors, both $\boldsymbol{f}_0^-$ and $\boldsymbol{f}_n^+$ are set to zero. All these forces are applied to the bones, which are simulated as rigid bodies, at the corresponding anchor points to drive the character's motion.

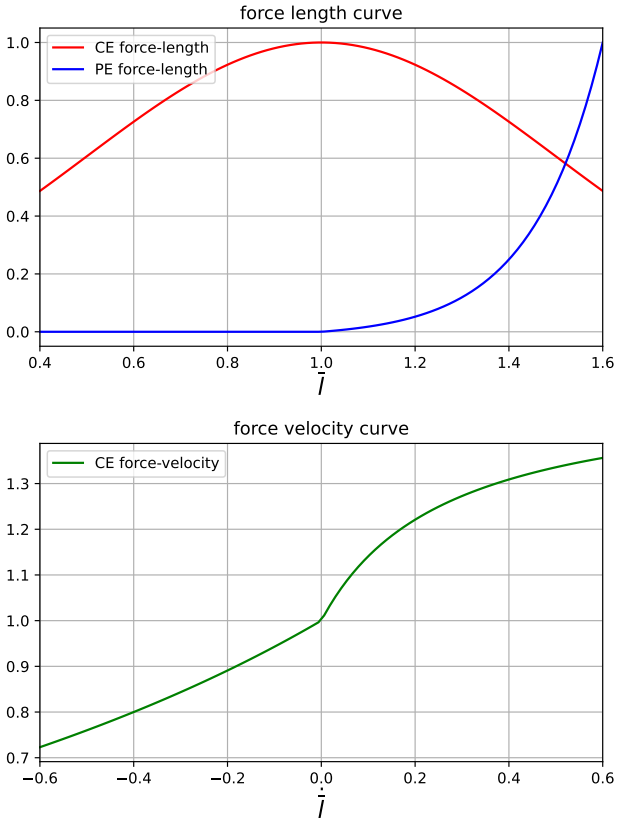

Figure 13: The force-length and force-velocity curves used in our experiment. Curves of the active muscle force are drawn in red and those of the passive muscle force are in blue.

A.2 Muscle Dynamics

The Hill-type muscle model [Hill 1938; Zajac 1989] is a widely adopted approach in character animation [Geijtenbeek et al. 2013; Lee et al. 2019]. This model comprises a contractile element (CE), a parallel elastic element (PE), and a tendon element. We simplify this model by neglecting changes in tendon length and the pennation angle following the work which are widely accepted in computer animation [Geijtenbeek et al. 2013; Jiang et al. 2019; Lee et al. 2019]. In the main article, we compute contractile muscle force as

$$f_{\mathrm{m}} = \alpha f_{\mathrm{CE}}^l(\bar{l}) f_{\mathrm{CE}}^v(\dot{\bar{l}}) + f_{\mathrm{PE}}(\bar{l}), \tag{21}$$

*corresponding author

SA Conference Papers '23, December 12–15, 2023, Sydney, NSW, Australia

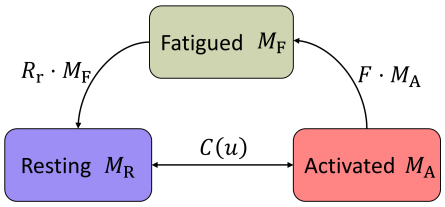

Figure 14: 3CC-r assumes muscle actuators to be in one of three possible states. These states are governed by a set of differential equations.

which is a compact version of the formulation

$$f_m = f_{m0}\left(\alpha \bar{f}^l_{CE}(\bar{l})\bar{f}^v_{CE}(\dot{\bar{l}}) + \bar{f}_{PE}(\bar{l})\right), \tag{22}$$

Here f_{m0} is the maximum isometric force, which is determined by the type, size, and several other properties of a muscle. The force-length and force-velocity functions, i.e., $\bar{f}^l_{CE}(\bar{l})$, $\bar{f}^v_{CE}(\dot{\bar{l}})$, and $\bar{f}_{PE}(\bar{l})$, are assumed to be the same for all the muscles. The normalized muscle length and its rate of change, $\bar{l}$ and $\dot{\bar{l}}$, are computed as

$$\bar{l} = \frac{l_M/l_{ori} - l_{Tnorm}}{l_{MTnorm}} \tag{23}$$

$$\dot{\bar{l}} = \frac{1}{\Delta t}\left(\bar{l}^t - \bar{l}^{t-1}\right), \tag{24}$$

where l_{ori} is the rest length of the muscle. l_{Tnorm} and l_{MTnorm} are the normalizing factors of tendon length and muscle-tendon unit length of the muscle, respectively. The values of f_{m0} and these normalizing factors for each muscle can be found in biomechanics literature, such as [Delp et al. 2007]. In this paper, we borrow these values from [Lee et al. 2019]. The formulation of the functions $\bar{f}^l_{CE}$, $\bar{f}^v_{CE}$, and $\bar{f}_{PE}$ used in this paper are:

$$\bar{f}^l_{CE}(\bar{l}) = \exp\left(-\frac{(\bar{l}-1)^2}{0.5}\right)$$

$$\bar{f}^v_{CE}(\dot{\bar{l}}) = \begin{cases} 1.5 + \dfrac{0.5 \times (-10.0 + \dot{\bar{l}})}{37.8\dot{\bar{l}} + 10.0} & \text{if } \dot{\bar{l}} > 0.0 \\ \dfrac{-10 - \dot{\bar{l}}}{-10.0 + 5.0\dot{\bar{l}}} & \text{otherwise} \end{cases}, \tag{25}$$

$$\bar{f}_{PE}(\bar{l}) = \begin{cases} \dfrac{\exp\left(\dfrac{4.0 \times (\bar{l}-1.0)}{0.6}\right) - 1.0}{\exp(4.0) - 1.0} & \text{if } \bar{l} > 1.0 \\ 0.0 & \text{otherwise} \end{cases}$$

Figure 13 shows the graphs of these functions.

A.3 Fatigue Dynamics

We adopt 3CC-r model [Looft et al. 2018] as the fatigue dynamics model, which is an enhanced version of the Three Compartment Controller (3CC) model proposed by Xia and Law [2008]. The 3CC-r model assumes that each muscle consists of multiple hypothetical muscle-tendon actuators. Each of these actuators is presumed to be in one of three possible states (compartments):

- **Activated** M_A: The muscle actuator is contributing.
- **Resting** M_R: The muscle actuator is inactivated but can be recruited.
- **Fatigued** M_F: The muscle actuator is fatigued and cannot be utilized.

We employ a unit-less measure of muscle force, expressed as a percentage of the maximum voluntary contraction (MVC), to describe the effect of fatigue, following existing literature. The values M_A, M_R, and M_F are expressed as percentages of MVC. The resting muscle actuator (M_R) is recruited to become an activated muscle actuator (M_A) when there is a load requirement. Once activated, the muscle actuator's power decays and fatigue accumulates. The transition relationships among the three states of the muscle are illustrated in Figure (14). The following equations describes the change of these values over time for each compartment:

$$\begin{aligned} \frac{dM_A}{dt} &= C(u) - F \cdot M_A \\ \frac{dM_R}{dt} &= -C(u) + R_r \cdot M_F \\ \frac{dM_F}{dt} &= F \cdot M_A - R_r \cdot M_F \end{aligned} \tag{26}$$

where F and R_r denote the fatigue and recovery coefficients. r is an additional rest recovery multiplier introduced by Looft et al. [2018], which alters the recovery coefficient as

$$R_r = \begin{cases} r \cdot R & M_A \geq u \\ R & M_A < u \end{cases} \tag{27}$$

The function $C(u)$ in Equation (26) dynamically change the ratio of M_A and M_R based on the target load u. It is formulated as a piecewise linear function that increases monotonically with u:

$$C(u) = \begin{cases} L_R \cdot (u - M_A) & \text{if } M_A \geq u \\ L_D \cdot (u - M_A) & \text{if } M_A < u \text{ and } M_R > u - M_A \\ L_D \cdot M_R & \text{if } M_A < u \text{ and } M_R \leq u - M_A, \end{cases} \tag{28}$$

which is characterized by the development factor L_D and relaxation factor L_R. It is worth noting that $C(u)$ also depends on the current activation level M_A. This relationship effectively prevents the activation level from changing instantaneously, thereby replicating the behavior of activation dynamics [Thelen 2003; Winters 1995].

The target load, u, represents the effort the brain expects the musculoskeletal system to generate, resulting from the combined effects of physiological and neurological processes. u can also be depicted as a normalized, unit-less coefficient that reflects the percentage of actuators a muscle is required to recruit, thus $u \in [0, 1]$. Furthermore, u effectively functions as the muscle excitation in the activation dynamics model [Thelen 2003; Winters 1995]. Notably, the target load u is allowed to change instantaneously within the range of $[0, 1]$. However, due to the existence of muscle fatigue, it may not always be realizable by the musculoskeletal system.

We provide an example in Figure 15 to illustrate the evolution of the components in the 3CC-r model. Initially, the muscle is in a non-fatigued and non-activated state with $M_A = M_F = 0.0$ and $M_R = 1.0$. The target load u is set to 0.5 for the first 20 seconds and then reset to 0. Note that Figure 15 merely serves as an illustration, the values of L_R, L_D, F, and R_r are adjusted to enhance the visibility of the curves.

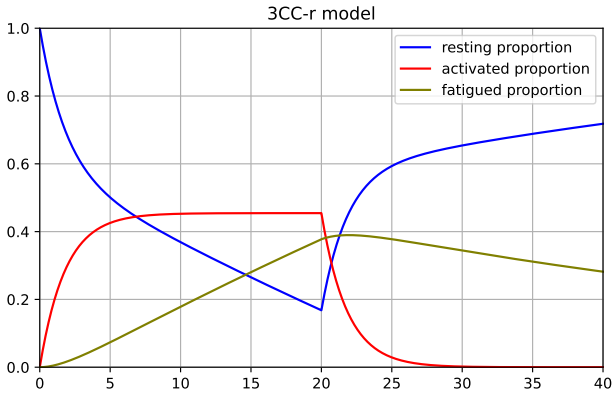

Figure 15: An example of 3CC-r to illustrate the evolution of M_A (red), M_R (blue), M_F (chartreuse) under the square wave target load stimuli. The muscle is set at non-fatigued and non-activated state at the beginning.

A.3.1 Fatigue dynamics as clip operation. Our muscle-space PD control calculates a desired force for each muscle. However, due to muscle dynamics and fatigue, these forces are not always achievable. As discussed in the main article, our strategy is to clip these desired forces to within feasible ranges and then apply the resulting forces to the character. Here, we derive the equations used to determine these feasible ranges.

The Hill-type muscle model given in Equation (21) suggests that the muscle force monotonically increases with respect to the activation level. We can then compute the desired muscle activation α_{pd} based on the desired force f_{pd} using

$$\alpha_{\text{pd}} = \frac{f_{\text{pd}} - f_{\text{PE}}(\bar{l})}{f^l_{\text{CE}}(\bar{l}) f^v_{\text{CE}}(\dot{\bar{l}})} \quad . \tag{29}$$

Notably, α_{pd} may not be achievable due to the muscle constraints and fatigue.

As discussed in the main article, M_A and α are equivalent because they both represent the muscle activation level. The first equation in Equation (26) can be rewritten as

$$\dot{\alpha} = C(u) - F\alpha, \tag{30}$$

which can be discretized using the forward Euler method. Denoting the muscle activation in the subsequent time step as $\tilde{\alpha}$, we have

$$\dot{\alpha} \approx \frac{\tilde{\alpha} - \alpha}{\Delta t}, \quad \text{or,} \quad \tilde{\alpha} \approx \dot{\alpha}\Delta t + \alpha. \tag{31}$$

$$\text{So,} \quad \tilde{\alpha} = \tilde{\alpha}(u) = K\alpha + \Delta t C(u), \tag{32}$$

where $K = 1 - \Delta t F$ can be considered as a decay factor. With Equation (32), our objective is now to find a target load u within its feasible range, $[0, 1]$, that can leads to a feasible $\tilde{\alpha}$ close to α_{pd}.

Substituting Equation (28) into Equation (32), we get

$$\tilde{\alpha}(u) = \begin{cases} K\alpha + \Delta t L_R (u - \alpha) & u \le \alpha \\ K\alpha + \Delta t L_D (u - \alpha) & \alpha < u < \alpha + M_R \\ K\alpha + \Delta t L_D M_R & u \ge \alpha + M_R. \end{cases} \tag{33}$$

It is easy to verify that $\tilde{\alpha}(u)$ is continuous and monotonically non-decreasing with respect to u. The minimum and maximum values of $\tilde{\alpha}(u)$, given the current activation level α, are:

$$\tilde{\alpha}_{\text{lb}} = \tilde{\alpha}(0) = \max(0, K\alpha - \Delta t L_R \alpha) \tag{34}$$

$$\tilde{\alpha}_{\text{ub}} = \tilde{\alpha}(1) = \min(1, K\alpha + \Delta t L_D M_R). \tag{35}$$

Here we use the facts that $\tilde{\alpha}, \alpha \in [0, 1]$ and $\alpha + M_R \in [0, 1]$. Now, we can calculate the feasible $\tilde{\alpha}$ that is close to α_{pd} using the clip operator

$$\tilde{\alpha}^* = \text{clip}(\alpha_{\text{pd}}, \tilde{\alpha}_{\text{lb}}, \tilde{\alpha}_{\text{ub}}) = \begin{cases} \tilde{\alpha}_{\text{lb}} & \alpha_{\text{pd}} \le \tilde{\alpha}_{\text{lb}} \\ \alpha_{\text{pd}} & \tilde{\alpha}_{\text{lb}} < \alpha_{\text{pd}} < \tilde{\alpha}_{\text{ub}} \\ \tilde{\alpha}_{\text{ub}} & \alpha_{\text{pd}} \le \tilde{\alpha}_{\text{ub}}. \end{cases} \tag{36}$$

Equivalently, we can clip the PD muscle force directly as described in Section 3.4.

After finding the feasible activation level $\tilde{\alpha}^*$, we can further calculate the corresponding u^* that leads to it and use u^* to simulate the 3CC-r model. However, considering that the governing equations in Equation (26) only depend on $C(u)$, we do not need to explicitly compute u^* but can compute $C(u^*)$ by inverting Equation (32). Specifically,

$$C(u^*) = \frac{\tilde{\alpha}^* - K\alpha}{\Delta t}, \tag{37}$$

which is used to update M_R in Equation (26) using the forward Euler method. In the meanwhile, M_F in Equation (26) is updated using the the current muscle activation α and M_F.

B MUSCLE VAE

B.1 Neural Network Structure

We formulate the components of the MuscleVAE, specifically the policy $\pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$, the posterior distribution $q(\boldsymbol{z}|\boldsymbol{s}, \tilde{\boldsymbol{s}}_{\text{skeleton}})$, and the state-dependent prior distribution $p(\boldsymbol{z}|\boldsymbol{s})$, as normal distributions in the form of $\mathcal{N}(\boldsymbol{\mu}_*(\bullet; \theta_*), \sigma_*^2 \boldsymbol{I})$. Here, σ_* is a predefined standard deviation, and the mean $\boldsymbol{\mu}_*(\bullet; \theta_*)$ is represented by a neural network with trainable parameters θ_*. We utilize a latent space $\mathcal{Z}$ with a dimension of 64 to encode both motion skills and fatigue style.

The state-conditional prior distribution is formulated as

$$p(\boldsymbol{z}|\boldsymbol{s}) \sim \mathcal{N}\left(\boldsymbol{\mu}_p(\boldsymbol{s}; \theta_p), \sigma_p^2 \boldsymbol{I}\right), \tag{38}$$

where $\sigma_p = 0.3$, $\boldsymbol{\mu}_p$ is a neural network with parameters θ_p. The posterior distribution $q(\boldsymbol{z}|\boldsymbol{s}, \tilde{\boldsymbol{s}}_{\text{skeleton}})$ is also a normal distribution

$$q(\boldsymbol{z}|\boldsymbol{s}, \tilde{\boldsymbol{s}}_{\text{skeleton}}) \sim \mathcal{N}\left(\hat{\boldsymbol{\mu}}_q, \sigma_q^2 \boldsymbol{I}\right) \tag{39}$$

We ensure $q(\boldsymbol{z}|\boldsymbol{s}, \tilde{\boldsymbol{s}}_{\text{skeleton}})$ to be close to the prior $p(\boldsymbol{z}|\boldsymbol{s})$ with the same standard deviation $\sigma_q = \sigma_p = 0.3$ and, following the technique used by ControlVAE [Yao et al. 2022], formulate the mean of the posterior distribution using a trainable offset function:

$$\hat{\boldsymbol{\mu}}_q = \boldsymbol{\mu}_p(\boldsymbol{s}) + \boldsymbol{\mu}_q(\boldsymbol{s}, \tilde{\boldsymbol{s}}_{\text{skeleton}}; \theta_q) \tag{40}$$

where θ_q represent a collection of neural network parameters. Notebly, with this formulation, the KL-divergence loss in Equation (15) of the main article has a simpler form:

$$\mathcal{L}_{\text{kl}} = \sum_{t=0} \gamma^t \left\| \boldsymbol{\mu}_q(\boldsymbol{s}^t, \tilde{\boldsymbol{s}}^{t+1}_{\text{skeleton}}) \right\|_2^2 / 2\sigma_p^2. \tag{41}$$

Both $\boldsymbol{\mu}_p$ and $\boldsymbol{\mu}_q$ are modeled using neural networks with two hidden layers consisting of 512 units each, and the Exponential Linear Unit (ELU) function as the activation function.

Similarly, we model the policy as a Gaussian distribution

$$\pi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{\mu}_\pi(\boldsymbol{s},\boldsymbol{z};\theta_\pi),\sigma_\pi^2\boldsymbol{I}) \tag{42}$$

where θ_π denotes the neural network parameters. We adopt a mixture-of-expert (MoE) structure consisting of six expert networks, each of which has three hidden layers with 512 units with ELU as activation function. The parameters of these experts are combined based on weights calculated by a gating network that includes two hidden layers of 64 units each. The standard deviation of policy distribution σ_π is set to 0.05.

The world model $\omega(\boldsymbol{s},\boldsymbol{a};\theta_w)$ is formulated as a deterministic neural network. It consists of four hidden layers, each with 512 units, and uses ELU activation functions. All of its parameters are collectively represented by θ_w. The world model outputs both the skeleton state and the fatigue state, with the latter representing a prediction of the fatigue state for the next time step. The handling of the skeleton state is similar to the methods described in [Fussell et al. 2021; Yao et al. 2022]. Notably, this world model is formulated in maximal coordinates. During the early stages of training, the model can sometimes produce inaccurate bone positions. Such inaccuracies often result in excessive muscle lengths, leading to significant passive muscle forces and causing unstable training. To mitigate this, we employ a differentiable forward kinematics procedure, leveraging the predicted local rotation to prevent infeasible bone positions.

B.2 Training

We employ the training algorithm from ControlVAE [Yao et al. 2022] to train our MuscleVAE model. In Brief, the training objective is to train the posterior distribution $q(\boldsymbol{z}|\boldsymbol{s},\tilde{\boldsymbol{s}}_{\text{skeleton}})$ and the policy $\pi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})$ to make the distribution of the generated motions $p(\tau)$ matches the distribution of a motion dataset $\mathcal{D}=\{\tilde{\tau}_i\}$. Here, the trajectory τ consists of a sequence of state $\{\boldsymbol{s}^t\}$ and, if available, the correspond action $\{\boldsymbol{a}^t\}$. Algorithm 1 outlines the major procedures of this algorithm. In this algorithm, $\mathcal{B}$ represents a buffer of simulation tuples, with each tuple consisting of a simulation state and its corresponding action. The parameters used in this training algorithm are set as follows: $N_B = 5\times10^4$, $N_B' = 2048$, $T_{\text{w}} = 8$, $T_{\text{VAE}} = 24$, and $N_{\text{batch}} = 512$.

B.3 High-Level Policy

Following [Yao et al. 2022], we formulate the task policy $\pi(\boldsymbol{z}^t|\boldsymbol{s}^t,\boldsymbol{g}^t)$ as a Gaussian distribution $\mathcal{N}(\hat{\boldsymbol{\mu}}_g,\sigma_g^2\boldsymbol{I})$ with a diagonal covariance $\sigma_g=\sigma_q$ and the mean function computed as

$$\hat{\boldsymbol{\mu}}_g = \boldsymbol{\mu}_p + \boldsymbol{\mu}_g(\boldsymbol{s}^t,\boldsymbol{g}^t;\theta_g) \tag{43}$$

where θ_g denotes the network parameters. We use a neural network with three hidden layers, each having 256 units, to model $\boldsymbol{\mu}_g$. The pseudocode for the training this task policy is outlined in Algorithm 2. The parameters $N_{\text{HL}} = 512$ and $T_{\text{HL}} = 16$ in our implementation.

Algorithm 1: Train MuscleVAE

Function Train( ):
  Initialize q, p, π, ω, $\mathcal{B} \leftarrow \varnothing$
  while not terminated **do**
    // collect simulation trajectories
    Remove the oldest N_B' simulation tuples from $\mathcal{B}$
    while $|\mathcal{B}| < N_B$ **do**
      Select $\tilde{\tau} = \{\tilde{\boldsymbol{s}}^0_{\text{skeleton}},\dots,\tilde{\boldsymbol{s}}^T_{\text{skeleton}}\}$ from $\mathcal{D}$
      $\boldsymbol{s}^0 \leftarrow [\tilde{\boldsymbol{s}}^0_{\text{skeleton}}, \text{random}(\boldsymbol{s}_{\text{fatigue}})]$
      $\tau \leftarrow$ **GenerateTrajectory**($\tilde{\tau}$, $\boldsymbol{s}^0$, q, π, None, $|\tilde{\tau}|$)
      Store τ and $\tilde{\tau}$ in $\mathcal{B}$
    end
    TrainWorldModel(ω, T_w, $\mathcal{B}$)
    TrainMuscleVAE(ω, q, p, π, T_{VAE}, $\mathcal{B}$)
  end
end
Function GenerateTrajectory($\tilde{\tau}$, $\boldsymbol{s}^0$, q, π, ω, T):
  $t \leftarrow 0$
  while not terminated *and* $t < T$ **do**
    if π *is* a list **then**
      Extract $\boldsymbol{a}^t$ from π
    else
      Extract $\tilde{\boldsymbol{s}}^{t+1}_{\text{skeleton}}$ from $\tilde{\tau}$
      Sample $\boldsymbol{z}^t \sim q(\boldsymbol{z}^t|\boldsymbol{s}^t,\tilde{\boldsymbol{s}}^{t+1}_{\text{skeleton}})$
      Sample $\boldsymbol{a}^t \sim \pi(\boldsymbol{a}^t|\boldsymbol{s}^t,\boldsymbol{z}^t)$
    end
    $\boldsymbol{s}^{t+1} \leftarrow \text{Simulate}(\boldsymbol{s}^t,\boldsymbol{a}^t)$ **if** ω is None **else** $\omega(\boldsymbol{s}^t,\boldsymbol{a}^t)$
    $t \leftarrow t+1$
  end
  $\tau = \{\boldsymbol{s}^0,\boldsymbol{a}^0,\boldsymbol{s}^1,\boldsymbol{a}^1,\dots,\}$
end
Function TrainWorldModel(ω, T, $\mathcal{B}$):
  $\mathcal{L} \leftarrow 0$
  for $i \leftarrow 0$ **to** N_{batch} **do**
    Sample $\tau^* = \{\boldsymbol{s}^0,\boldsymbol{a}^0,\boldsymbol{s}^1,\boldsymbol{a}^1,\dots,\}$ from $\mathcal{B}$, ignore $\tilde{\tau}^*$
    $\pi^* \leftarrow \{\boldsymbol{a}^0,\boldsymbol{a}^1,\boldsymbol{a}^2,\dots\}$
    $\bar{\tau} \leftarrow$ **GenerateTrajectory**(None, $\boldsymbol{s}^0$, None, π^*, ω, T)
    $\mathcal{L} \leftarrow \mathcal{L} + \mathcal{L}_{\text{w}}(\bar{\tau},\tau^*)$
  end
  Update ω with $\mathcal{L}_{\text{w}}$
end
Function TrainMuscleVAE(ω, q, p, π, T, $\mathcal{B}$):
  $\mathcal{L} \leftarrow 0$
  for $i \leftarrow 0$ **to** N_{batch} **do**
    Sample τ^* and $\tilde{\tau}^*$ from $\mathcal{B}$
    Extract $\boldsymbol{s}^0$ from τ^*
    $\tau \leftarrow$ **GenerateTrajectory**($\tilde{\tau}^*$, $\boldsymbol{s}^0$, q, π, ω, T)
    $\mathcal{L} \leftarrow \mathcal{L} + \mathcal{L}_{\text{rec}}(\tau,\tilde{\tau}^*) + \beta\mathcal{L}_{kl}(\tau) + \mathcal{L}_{\text{act}}(\tau)$
  end
  Update q, p, π with $\mathcal{L}$
end

Algorithm 2: Train High-Level Policy

Function TrainVelocityControl(p, ω, π) :
  Initialize $\mathcal{B}$ with random simulated trajectories $\{\tau\}$
  $\mathcal{L}_{\boldsymbol{g}} \leftarrow 0$
  for $i \leftarrow 0$ **to** N_{NL} **do**
    Select a random task $\hat{\boldsymbol{g}}$
    Sample $\boldsymbol{s}^0$ from $\mathcal{B}$
    for $t \leftarrow 0$ **to** T_{NL} **do**
      Compute task parameter $\boldsymbol{g}^t$ according to $\boldsymbol{s}^t$ and $\hat{\boldsymbol{g}}$
      Sample $\boldsymbol{z} \sim \pi_{\boldsymbol{g}}(\boldsymbol{z}|\boldsymbol{s}^t, \boldsymbol{g}^t)$
      Sample $\boldsymbol{a}^t \sim \pi(\boldsymbol{a}^t|\boldsymbol{s}^t, \boldsymbol{z}^t)$
      $\boldsymbol{s}^{t+1} \leftarrow \omega(\boldsymbol{s}^t, \boldsymbol{a}^t)$
      $t \leftarrow t + 1$
    end
    $\tau_{\boldsymbol{g}} \leftarrow \{\boldsymbol{s}^t, \boldsymbol{z}^t\}$
    $\mathcal{L}_{\boldsymbol{g}} \leftarrow \mathcal{L}_{\boldsymbol{g}} + \mathcal{L}_{\boldsymbol{g}}(\tau_{\boldsymbol{g}})$
  end
  Update $\pi_{\boldsymbol{g}}$ with $\mathcal{L}_{\boldsymbol{g}}$
end

The loss functions have the form

$$\mathcal{L}(\tau_{\boldsymbol{g}}) = \sum_{t=1}^{T} \left[\mathcal{L}_{\boldsymbol{g}}(\boldsymbol{s}^t) + \mathcal{L}_{\text{fall}}(\boldsymbol{s}^t)\right] + w_z \sum_{t=0}^{T-1} \|\boldsymbol{\mu}_{\boldsymbol{g}}\|_2^2, \tag{44}$$

where $\mathcal{L}_{\boldsymbol{g}}(\boldsymbol{s}^t)$ is the task-specific objective function. The $\mathcal{L}_{\text{fall}}$ term penalizes falling down. The regularization term ensures that the mean value shift between the goal prior and the fixed low-level prior remains low, ensuring the minimal change in motion quality.

We use the heading control task in [Yao et al. 2022] to test MuscleVAE. In this task, the character is required to move in a specific direction indicated by the target direction $\theta_h \in [-\pi, \pi]$ at a given speed of $v \in [0.0, 3.0]$ m/s. The objective function for this task is defined based on the character's accuracy to reach its target direction while maintaining the specified speed. Specifically,

$$\mathcal{L}_{\boldsymbol{g}}(\boldsymbol{s}) = w_{\theta_h}|\theta_h^* - \theta_h| + w_v \frac{|v^* - v|}{\max(v^*, 1)}, \tag{45}$$

where θ_h and v are the character's current heading direction and velocity, respectively. θ_h^* is the target heading direction and v^* is the target velocity. $w_{\theta_h} = 2.0$ and $w_v = 1.0$ are balancing weights.

B.4 Other Implementation Details

Fatigue State. The fatigue state of the character is characterized by the values of M_{A}, M_{F}, and M_{R} for all the muscles. However, naively stacking all these variables into a single vector would lead to a very high-dimensional representation. To address this, we employ a more compact representation. We categorize all the muscles into five parts corresponding to the trunk and the four limbs. The fatigue state of the character, $\boldsymbol{s}_{\text{fatigue}}$, is then defined by the average value of M_{A}, M_{F}, and M_{R} for these five parts. We use a weighted average strategy to compute these values. In this approach, the fatigue parameters of each muscle are weighted by their maximum isometric force, f_{m0}. Since f_{m0} is typically larger for major muscles, this strategy ensures that the fatigue states of the major muscles have a greater impact on the policy.

Fatigue Initialization. During training, we initialize the fatigue variables M_{A}, M_{F}, and M_{R} randomly each time the environment resets. To ensure a valid combination of these variables, we select a random point from a predetermined fatigue evolution curve, which is generated by tracking a synthetic target load pattern. A typical curve for this initialization is illustrated in Figure 15.

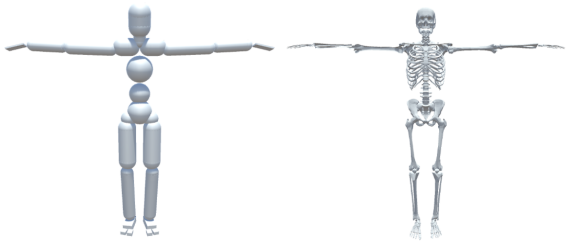

Figure 16: The physics collision geometries (left) and rendering mesh (right) of our character. The physics-based character is composed of 23 rigid bodies interconnected by 22 joints. We set the elbows and knees as hinge joints, while the other joints are set as ball-socket joints.

C EXPERIMENTS

C.1 Character

The character model depicted in Figure 16 is used in all our experiments. It has a height of 1.68 m, weighs 61.4 kg, consists of 23 rigid bodies connected by 22 joints, and is actuated by 284 muscles. The muscle model, including both the muscle dynamics and fatigue dynamics, operates at a frequency of 120 Hz. We implement the implicit joint damping mechanism to ensure the numerical stability of the simulation with a large timestep. The damping coefficient $k_{\text{d-joint}} = 10.0$ is applied uniformly to all joints. For each muscle, the stiffness parameter k_{p} is set to the same value as the Hill-type maximum isotropic force, and the damping coefficient k_{d} is set to $0.1k_{\text{p}}$. The original 3CC-model paper [Xia and Law 2008] suggests that there are three types of muscles: slow (S), fatigue-resistant (FR), and fast fatigue (FF). As a simplified model, we assume all the muscles are S-muscles. The parameters of fatigue are then $F = 0.01$, $R = 0.002$, $L_{\text{D}} = L_{\text{R}} = 50.0$ and $r = 2.0$. We also test other fatigue ratio in the experiment showed at the last paragraph of Section 5.2. We keep the ratio of F over R at 5 for all experiments except the arm holding experiment, where $F/R = 20, F = 0.1$ for faster reaching the powerless posture of the arms.

C.2 Dataset

Table 1 lists the motions used in our experiments. All these motions are selected from the LaFAN dataset [Harvey et al. 2020]. The last row of Table 1 denotes the unseen motion clip of 8th demo in the supplementary video which is only used in testing rather than training. We use the dance motion from [Lee et al. 2019] for testing which is the 9th demo in the supplementary video. The motion data

Table 1: Motions Used for the Locomotion MuscleVAE

Motion	Frames (20 fps)
Walk	5227
Run	4757
Jump	4889
Run2(Test)	5477

of *Jump Spin Kick* and *Horse Stance* has already been mentioned in the main text.

C.3 Muscle Render

To more clearly reflect the muscle activation state, we use a linear relationship from white to red, with red indicating muscles that are more activated. Simultaneously, we increase the width of the muscle polylines in our visualization for muscles with higher activation levels.

REFERENCES

Scott L Delp, Frank C Anderson, Allison S Arnold, Peter Loan, Ayman Habib, Chand T John, Eran Guendelman, and Darryl G Thelen. 2007. OpenSim: open-source software to create and analyze dynamic simulations of movement. *IEEE transactions on biomedical engineering* 54, 11 (2007), 1940–1950.

Levi Fussell, Kevin Bergamin, and Daniel Holden. 2021. SuperTrack: Motion Tracking for Physically Simulated Characters Using Supervised Learning. *ACM Trans. Graph.* 40, 6, Article 197 (dec 2021), 13 pages. https://doi.org/10.1145/3478513.3480527

Thomas Geijtenbeek, Michiel Van De Panne, and A Frank Van Der Stappen. 2013. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–11.

Félix G Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. 2020. Robust motion in-betweening. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 60–1.

Archibald Vivian Hill. 1938. The heat of shortening and the dynamic constants of muscle. *Proceedings of the Royal Society of London. Series B-Biological Sciences* 126, 843 (1938), 136–195.

Yifeng Jiang, Tom Van Wouwe, Friedl De Groote, and C Karen Liu. 2019. Synthesis of biologically realistic human motion using joint torque actuation. *ACM Transactions On Graphics (TOG)* 38, 4 (2019), 1–12.

Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. 2019. Scalable Muscle-Actuated Human Simulation and Control. *ACM Trans. Graph.* 38, 4, Article 73 (jul 2019), 13 pages. https://doi.org/10.1145/3306346.3322972

John M Looft, Nicole Herkert, and Laura Frey-Law. 2018. Modification of a three-compartment muscle fatigue model to predict peak torque decline during intermittent tasks. *Journal of biomechanics* 77 (2018), 16–25.

Darryl G. Thelen. 2003. Adjustment of Muscle Mechanics Model Parameters to Simulate Dynamic Contractions in Older Adults. *Journal of Biomechanical Engineering* 125, 1 (Feb. 2003), 70–77. https://doi.org/10.1115/1.1531112

Jack M. Winters. 1995. An Improved Muscle-Reflex Actuator for Use in Large-Scale Neuromusculoskeletal Models. *Annals of Biomedical Engineering* 23, 4 (July 1995), 359–374. https://doi.org/10.1007/BF02584437

Ting Xia and Laura A Frey Law. 2008. A theoretical approach for modeling peripheral muscle fatigue and recovery. *Journal of biomechanics* 41, 14 (2008), 3046–3052.

Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. 2022. ControlVAE. *ACM Transactions on Graphics* 41, 6 (nov 2022), 1–16. https://doi.org/10.1145/3550454.3555434

Felix E Zajac. 1989. Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. *Critical reviews in biomedical engineering* 17, 4 (1989), 359–411.