Proceedings of Machine Learning Research vol vvv:1–8, 2025

Supplemental Material
BIGE : Biomechanics-informed GenAI for Exercise Science

Shubh Maheshwari SHMAHESHWARI@UCSD.EDU
Anwesh Mohanty ANMOHANTY@UCSD.EDU
Yadi Cao YAC066@UCSD.EDU
Swithin Razu SRAZU@UCSD.EDU
Andrew McCulloch AMCCULLOCH@UCSD.EDU
Rose Yu ROSEYU@UCSD.EDU
University of California, San Diego
La Jolla, CA 92093

Contents

1. Data processing

1.1. Temporal Segmentation

Each sample in our reference dataset consists of an athlete performing an exercise multiple times (3–4 repetitions per sample). To process this data, we temporally segment individual motion iterations from the original recordings. The start and end frames for each motion segment are identified using the angular velocity of the joints.

Figure 1 illustrates this process, showing the angular velocity of various joints (in degrees per second) over time for an athlete performing squats. The red points in the bottom plot in Figure 1 indicate the timestamps where each squat iteration ends. For squats specifically, we used the maximum angular velocity of the *left and right knee flexion* to determine the start and end points of each motion. The closest matching time points from both knees were combined to ensure accurate segmentation.

Each sample in our reference dataset consists of an athlete performing an exercise multiple times (3–4 repetitions per sample). To process this data, we temporally segment individual motion iterations

© 2025 S. Maheshwari, A. Mohanty, Y. Cao, S. Razu, A. McCulloch & R. Yu.

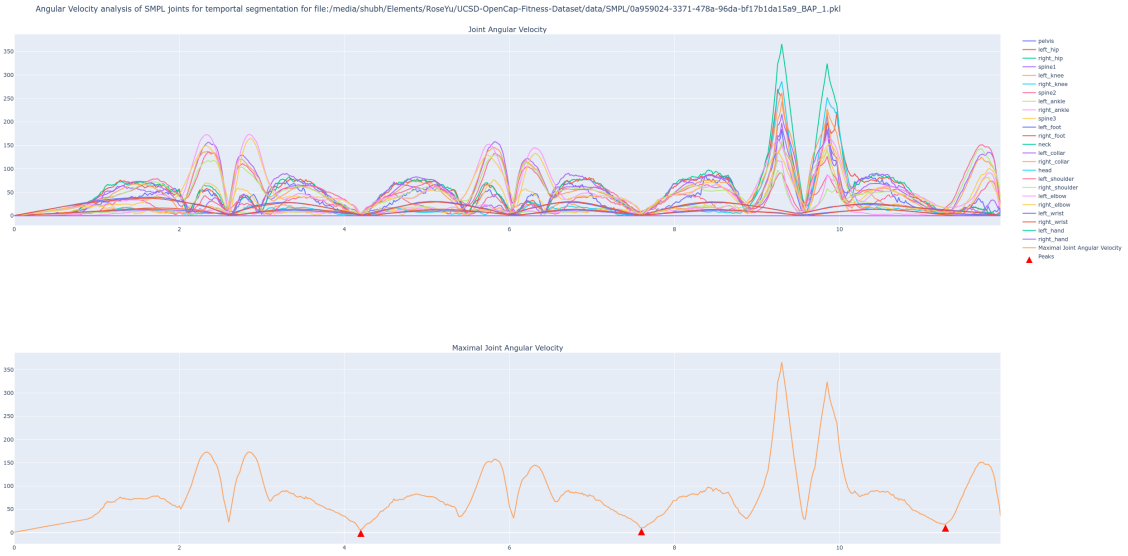

Figure 1: The top plot in the image shows the angular velocity for each joint (in deg/s) at each time step for an athlete doing squats. The red markers in the bottom plot indicate the time step at which the activity ends.

from the original recordings. The start and end frames for each motion segment are identified using the angular velocity of the joints.

Figure 1 illustrates this process, showing the angular velocity of various joints (in degrees per second) over time for an athlete performing squats. The red points arrows in the bottom plot of Figure 1 indicate the change points, the timestamps where each squat iteration ends. The motion was captured at 60 fps.

To automatically detect the start-stop cycle of each squat, first we apply savgol filterLuo et al. (2005) with a window length of 21 frames or (0.35 secs) with an order of 3 to smoothen the data. Next to determine the candidates for start and end points of each motion, we identify the valleys in the maximum angular velocity of the *left and right knee flexion*. We choose only from the bottom 10% percent and allow a distance of 6 frames or (0.1 seconds) between the candidates. Then for all best of 3 combinations of start and stop pairs, we perform cubic interpolation to normalize the segmented motion to 101 timesteps. Lastly, we choose the 3 pairs with the least standard deviation. We also penalize combinations with average length greater than 2 seconds.

1.2. Muscle Simulation Setup

OpenCap Uhlrich et al. (2022) can estimate dynamics using muscle-driven tracking simulations of joint kinematics. The tracking simulations are formulated as optimal control problems that aim to identify muscle excitations that minimize a cost function subject to constraints describing muscle and skeletal dynamics. The cost function J (Equation 1) includes squared terms for muscle activations (a) and excitations of the ideal torque motors at the lumbar, shoulder, and elbow joints (e_{tm}). It also includes tracking terms (squared differences between simulated and reference data), namely tracking of experimental joint positions ($\boldsymbol{q}$), joint velocities ($\dot{\boldsymbol{q}}$), and joint accelerations ($\ddot{\boldsymbol{q}}$):

The musculoskeletal model is driven by 80 muscles actuating the lower-limb coordinates and 13 ideal torque motors actuating the lumbar, shoulder, and elbow coordinates. Ground reaction forces (i.e., external forces) are modeled through six-foot-ground contact spheres attached to the foot

segments of the model. Raasch's model is used to describe muscle excitation-activation coupling and a Hill-type muscle model is used to describe muscle-tendon dynamics and the dependence of muscle force on muscle fiber length and velocity. Skeletal motion is modeled with Newtonian rigid body dynamics and smooth approximations of compliant Hunt-Crossley foot-ground contacts.

$$
\begin{aligned}
J = w_1 \int_{t_0}^{t_f} \sum_{i=1}^{N_m} a_i^2(t)\, dt + w_2 \int_{t_0}^{t_f} \sum_{j=1}^{N_{\text{tm}}} e_{\text{tm},j}^2(t)\, dt \\
+ w_3 \int_{t_0}^{t_f} \|\boldsymbol{q}(t) - \boldsymbol{q}_{\text{ref}}(t)\|^2\, dt + w_4 \int_{t_0}^{t_f} \|\dot{\boldsymbol{q}}(t) - \dot{\boldsymbol{q}}_{\text{ref}}(t)\|^2\, dt \\
+ w_5 \int_{t_0}^{t_f} \|\ddot{\boldsymbol{q}}(t) - \ddot{\boldsymbol{q}}_{\text{ref}}(t)\|^2\, dt
\end{aligned}
\tag{1}
$$

where t_0 and t_f are the initial and final times, w_i with $i = 1, \ldots, 5$ are weights, t is time, N_m is the number of muscles, and N_{tm} is the number of torque motors. Experimental joint positions, velocities, and accelerations are low-pass filtered using fourth-order, zero-lag Butterworth filters (default cutoff frequencies are 12 Hz for gait trials and 30 Hz for non-gait trials). Each cost term is scaled with empirically determined weights. A penalty function is appended to the cost function with the remaining control variables to avoid singular arcs. Note that the optimal control problem formulation can be tailored to the activity of interest to incorporate activity-based knowledge by, for instance, adjusting the cost function, constraints, and filter settings. The optimal control problems are formulated in Python with CasADi (v3.5), using direct collocation and implicit formulations of the muscle and skeletal dynamics. Algorithmic differentiation is used to compute derivatives. IPOPT is used to solve the resulting nonlinear programming problems with a convergence tolerance of 1×10^{-3} (all other settings are kept at default).

2. Surrogate model

To predict muscle activations efficiently, we developed a surrogate model inspired by the approach in Werling et al. (2023), using a 3-layer transformer architecture with 3 attention heads per layer. The model captures temporal dependencies in the input motion data while approximating the muscle activation outputs generated by physics-based simulations.

The ground-truth data for training the surrogate model was generated by processing the reference dataset through the OpenCap muscle simulation pipeline, as detailed in Section 1.2. The input to the model consists of joint kinematics from the reference dataset with dimensions 196×33 (196 time-steps, 33 degrees of freedom), and the output comprises the corresponding muscle activations with dimensions 196×80 (196 time-steps, 80 muscle-tendon actuators). The model's internal feature dimension was set to 128 to balance expressiveness and computational efficiency.

The surrogate model was trained with a dropout rate of 0.1 to prevent overfitting and an initial learning rate of 0.0002. The training was performed for 1000 epochs using the Adam optimizer Joo et al. (2018) with a batch size of 128. The learning rate was reduced by a factor of 0.5 whenever the validation loss plateaued for 10 consecutive epochs. The training loss was computed as the L1-smooth loss between the predicted and ground-truth muscle activations. Additionally, gradient clipping with a maximum norm of 1.0 was applied to stabilize training, and weight decay of 10^{-5} was used to regularize the model parameters.

Table 1 reports the performance of the trained surrogate model in predicting muscle activations for squat motions. Metrics include R^2 scores and root mean squared error (RMSE) for key muscles of interest. These results demonstrate the surrogate model's accuracy in approximating the outputs of computationally expensive simulations.

Metrics	R2	RMSE
Soleus (Left)	0.639	0.117
Soleus (Right)	0.614	0.119
Vastus Intermedius (Left)	0.665	0.068
Vastus Intermedius (Right)	0.676	0.063
Vastus Lateralis (Left)	0.873	0.112
Vastus Lateralis (Right)	0.799	0.134
Vastus Medialis (Left)	0.727	0.089
Vastus Medialis (Right)	0.679	0.092

Table 1: Metrics for Surrogate

3. Experiment Details

3.1. Baselines

For comparison with BIGE, samples were generated from MDM Tevet et al. (2022) and T2M-GPT Zhang et al. (2023) by prompting them to generate suitable squatting motion. Using NimblePhysics[1] (Werling et al. (2021)) the generated motion was retargeted to the joint kinematics format of Lai et al. (2017) musculoskeletal model.

Prompting: For both models, the phrase "person doing squats with hands behind head" worked best and led to the most accurate motion. Pre-trained models were used for both baselines to generate 20 samples for 'squat' motion.

Inverse Kinematics: Using individual athlete information (height, bone scale etc), inverse kinematics was applied to convert the data from HumanML3D (Guo et al. (2022)) format $\mathcal{H}$ into joint kinematics format K.

MDM and T2M-GPT utilize the joint representation proposed in Guo et al. (2022) where the pose at each instant is represented as a 263-dimensional pose vector composed of various components: $\mathbf{p} = [\mathbf{r}_r, \mathbf{r}_l, \mathbf{r}_y, \mathbf{r}_i, \mathbf{r}_o, \mathbf{v}_l, \mathbf{f}_c]$. For N samples with T time-steps, $\mathbf{r}_r \in \mathbb{R}^{T\times N\times 1}$ represents the root rotation velocity, $\mathbf{r}_l \in \mathbb{R}^{T\times N\times 2}$ is the root linear velocity, $\mathbf{r}_y \in \mathbb{R}^{T\times N\times 1}$ is the y-coordinate of the root, $\mathbf{r}_i \in \mathbb{R}^{T\times N\times (J-1)\times 3}$ is the translation of the remaining $J-1$ joints (with respect to the root joint), $\mathbf{r}_o \in \mathbb{R}^{T\times N\times (J-1)\times 6}$ is the rotation data for the remaining $J-1$ joints, $\mathbf{v}_l \in \mathbb{R}^{T\times N\times J\times 3}$ is the local velocity for all J joints, and $\mathbf{f}_c \in \mathbb{R}^{T\times N\times 4}$ is the foot contact information. Here $J = 22$ and represent the first 22 joints of SMPL (Loper et al. (2015)).

The root joint is initialized at origin. At time-step t, first using r_r, the orientation of root joint is calculated as:

$$R^t = \left[\cos(\textstyle\sum_{i=0}^{t} r^i_{r,0}), 0, \cos(\sum_{i=0}^{t} r^i_{r,1}), 0\right]^T$$

1. https://nimblephysics.org/docs/markers-and-joint-centers.html#converting-motion-between-skeleton-types

The location of the root joint is obtained by cumulating the translation of root joint $\mathcal{H}^t_{root} = \mathcal{H}^t_{root} + R^t - 1 \cdot r^t_l$ The location of the remaining j joints, is calculated as $\mathcal{H}^t_j = R^t \cdot r^t_{i,j}$. To obtain joint kinematics K^t, we use inverse kinematics (IK) procedure in NimblePhysics using $\mathcal{H}$ as markers until the difference in error rate converges to less than 10^{-4}.

3.2. Hyperparameters

Table 2 the hyperparameters used in the training. During guidance we perform a cold start using only the proximity loss for the first 500 iterations and then gradually introducing all the guidance parameters α mentioned in Table. 3. Again at the end for the last 500 iterations we only apply the proximity loss.

Argument	Value	Description
`total-iterations`	300000	Total iterations for training
`batch-size`	128	Batch-size for training
`learning-rate`	0.0002	Initial learning rate for training
`lr-scheduler`	50000	Modify learning rate after every 50000 steps
`gamma`	0.05	Factor to modify learning rate
`loss-commitment`	0.02	Commitment loss of VQVAE
`loss-penetration`	0.05	Penetration loss of VQVAE
`embedding-dimension`	512	Dimension of the embedding space of VQVAE
`nb-codes`	512	Number of notebook vectors for VQVAE

Table 2: Hyperparameters for VQVAE training

Argument	Value	Description
`total-iterations`	3000	Total iterations for guidance
`batch-size`	20	Number of samples to generate
β	0.01	Proximity regularizer
α_{tilt}	0.5	Guidance parameter to reduce pelvis tilt.
α_{asymm}	1	Guidance parameter to reduce asymmetry
α_{ω}	0.5	Hyperparameter for foot regularizer
$\alpha_{v_{\text{COM}}}$	50	Guidance parameter to reduce pelvis jitteriness
$\alpha_{a_{\text{COM}}}$	100	Guidance parameter to reduce pelvis acceleration
α_{slide}	0.1	Guidance parameter to reduce foot sliding
α_{MAC}	1	Guidance parameter for muscle activation constrain

Table 3: Hyperparameters for guidance

3.3. Qualitative Comparison

We provide a frame-by-frame comparison of BIGE, MDM and VQVAE over a single squat cycle in Figure 2. The motions generated by MDM and VQVAE fail to achieve sufficient depth in the squat and exhibit unnatural artifacts during the cycle. In contrast, BIGE produces motions closely aligned

with the reference data and free of such artifacts. The biomechanical constraints enforced in BIGE enable the model to perform deep, natural squats throughout the cycle.

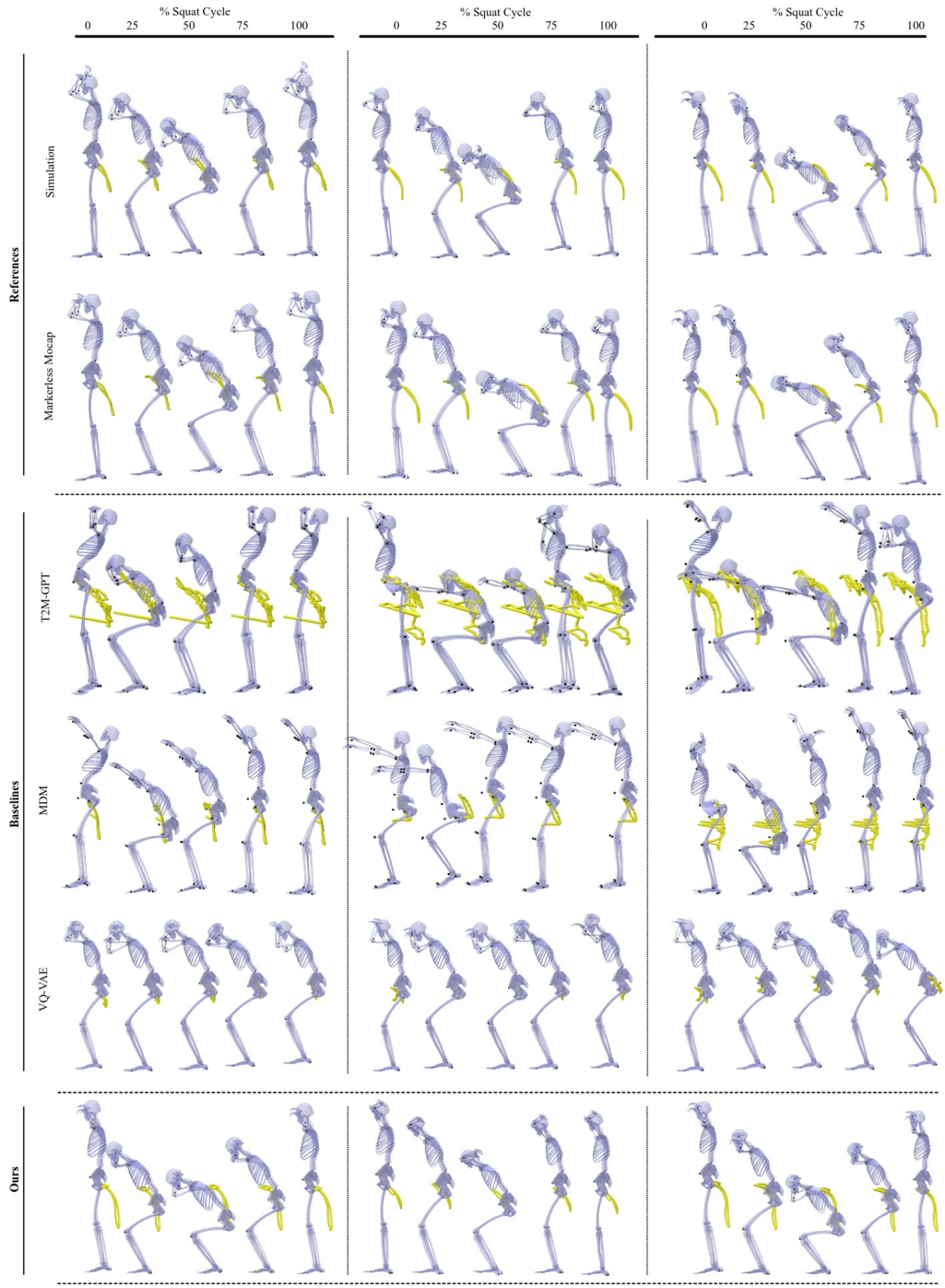

Figure 2: **Qualitative Results:** Comparison of generated samples of BIGE with references and baselines. The yellow curve represents the movement of the hip joint over the entire squat cycle. BIGE generates a more natural squat motion compared to MDM and T2M-GPT.

4. Computation Cost

Our experiments were conducted using a local cluster equipped with 8 Nvidia RTX 2080 Ti GPUs, each with 12GB of memory, an Intel Xeon processor, and 128GB of RAM. This setup enabled efficient training and inference for all components of our pipeline.

To train the VQ-VAE, we utilized DeepSpeedRasley et al. (2020) to distribute the workload across all 8 GPUs, allowing for larger batch sizes and faster convergence. The VQ-VAE model required approximately 24 hours to train. The surrogate model, on the other hand, was lightweight and trained in just 30 minutes on a single GPU.

For inference, generating motion sequences with a batch size of 64 took approximately 3 seconds on a single GPU, demonstrating the computational efficiency of our approach. GPU memory utilization peaked at 10GB per GPU during training, while inference tasks required significantly less memory, making the framework practical for real-time or batch applications.

While training the VQ-VAE remains resource-intensive due to the size of the latent space and motion data, the use of DeepSpeed and distributed training substantially reduced the computational burden. Future optimizations, such as model pruning and more efficient architectures, could further enhance scalability and reduce training costs.

References

Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5152–5161, 2022.

Hanbyul Joo, Tomas Simon, and Yaser Sheikh. Total capture: A 3d deformation model for tracking faces, hands, and bodies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Adrian Lai, Allison Arnold, and James Wakeling. Why are antagonist muscles co-activated in my simulation? a musculoskeletal model for analysing human locomotor tasks. *Annals of Biomedical Engineering*, 45, 09 2017. doi: 10.1007/s10439-017-1920-7.

Matthew Loper, Naureen Mahmood, et al. Smpl: A skinned multi-person linear model. *SIGGRAPH Asia*, 34(6):248:1–248:16, 2015.

Jianwen Luo, Kui Ying, and Jing Bai. Savitzky–golay smoothing and differentiation filter for even number data. *Signal Processing*, 85(7):1429–1434, 2005. ISSN 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2005.02.002. URL https://www.sciencedirect.com/science/article/pii/S0165168405000654.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3406703. URL https://doi.org/10.1145/3394486.3406703.

Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H Bermano. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022.

Scott Uhlrich, Antoine Falisse, Łukasz Kidziński, Julie Muccini, Michael Ko, Akshay Chaudhari, Jennifer Hicks, and Scott Delp. Opencap: 3d human movement dynamics from smartphone videos. 07 2022. doi: 10.1101/2022.07.07.499061.

Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and Karen Liu. Fast and feature-complete differentiable physics engine for articulated rigid bodies with contact constraints. 07 2021. doi: 10.15607/RSS.2021.XVII.034.

Keenon Werling, Nicholas A. Bianco, Michael Raitor, Jon Stingel, Jennifer L. Hicks, Steven H. Collins, Scott L. Delp, and C. Karen Liu. Addbiomechanics: Automating model scaling, inverse kinematics, and inverse dynamics from human motion data through sequential optimization. *PLOS ONE*, 18(11):1–25, 11 2023. doi: 10.1371/journal.pone.0295152. URL https://doi.org/10.1371/journal.pone.0295152.

Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Shaoli Huang, Yong Zhang, Hongwei Zhao, Hongtao Lu, and Xi Shen. T2m-gpt: Generating human motion from textual descriptions with discrete representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.