XMimic: Learning Whole-Body Motion Tracking in Under a Minute

Nico Bohlinger[1], Jan Peters[1,2]

***Abstract*— We present XMimic, which learns whole-body motion tracking for the Unitree G1 humanoid in under a minute. XMimic reimplements BlinkMimic, a fast open-source version of SuperMimic, in JAX and runs its complete training loop on the GPU. It compiles every training iteration into a single GPU program and can split the iteration across up to four GPUs that work on it at the same time. On four H100 GPUs, XMimic learns to track all 40 LAFAN1 motion clips in 0.91 minutes, 3.5 times faster than BlinkMimic with the same algorithm and hyperparameters. The code is available at https://github.com/nico-bohlinger/xmimic.**

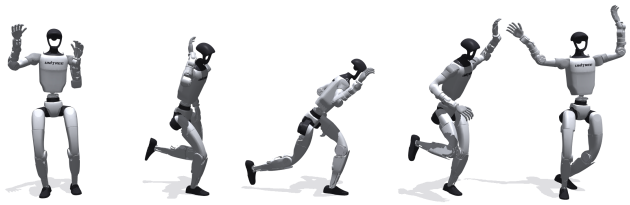


Fig. 1. We train a single policy that tracks all 40 LAFAN1 motion clips with the Unitree G1 in under a minute on four H100 GPUs. The render shows the policy after 54 seconds of training at five moments of a dance clip.

I. INTRODUCTION

Tracking human motion data has become a common way to give humanoid robots natural whole-body skills, where a single policy learns to imitate a large set of motion clips in simulation [1]. SuperMimic [2] learns such a policy with a world model, which is a neural network that predicts how the robot moves. Instead of learning only from the rewards of trial and error, the policy improves by backpropagating the tracking reward through motions that it imagines with the world model, similar to SuperTrack [3] for simulated characters. The community quickly took up this idea. Starting from an unofficial reimplementation [4], InstantTrack [5] and BlinkMimic [6] reduced the time to learn all 40 clips of the LAFAN1 dataset [7] on the Unitree G1 from about 1.5 hours to about 3 minutes on a single consumer GPU. However, these implementations no longer get faster with better hardware, as BlinkMimic trains no faster on an NVIDIA H100 than on an RTX 4090. The reason is that every training iteration consists of many small GPU operations that Python starts one after another, so the GPU often waits for Python instead of computing. Additionally, a single GPU has to run all parts of an iteration one after another, so more GPUs do not shorten the training.

[1]Department of Computer Science, Technical University of Darmstadt, Germany. [2]Robotics Institute Germany (RIG); German Research Center for AI (DFKI); hessian.AI.
Corresponding author: nico.bohlinger@tu-darmstadt.de

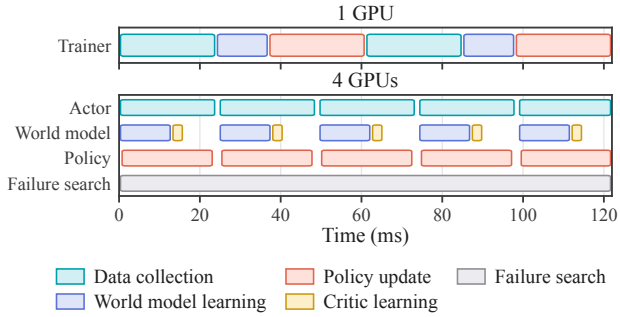


Fig. 2. Training iterations on one GPU (top) and split across four GPUs (bottom), drawn to scale with times measured on H100 GPUs. On four GPUs, the actor, the world-model learner, and the policy learner work at the same time, each with the results of the others from the previous iteration, while the fourth GPU runs the failure search.

This is why we propose XMimic, which reimplements BlinkMimic in JAX [8], a Python library that compiles numerical code, and runs the complete training loop on the GPU, following the idea of InstantMimic [9]. XMimic compiles every training iteration into a single GPU program and, with more GPUs, splits the iteration across up to four GPUs that work on its parts at the same time. With the same algorithm and hyperparameters as BlinkMimic, XMimic learns to track all 40 LAFAN1 clips in 0.91 minutes on four H100 GPUs, which is 3.5 times faster than BlinkMimic, and it is already 1.3 times faster on a single GPU.

II. MOTION TRACKING IN UNDER A MINUTE

We build upon BlinkMimic [6] and keep its algorithm and hyperparameters, so that all speedups come from how the training runs on the hardware. BlinkMimic trains the policy in iterations that consist of three steps (see Figure 2). First, the policy collects new experience in 4096 parallel environments of the simulator, where the robot tries to follow the motion clips. Second, the world model learns to predict the next state of the robot from its current state and action, using the experience in the replay buffer. Third, the policy imagines the next 32 steps of its motion with the world model and improves by backpropagating the tracking reward through these imagined steps, while a critic estimates the reward that follows after them. Additionally, BlinkMimic regularly plays the current policy on all motion clips to find the moments where the robot still falls and starts part of the training episodes shortly before them, which we call the failure search.

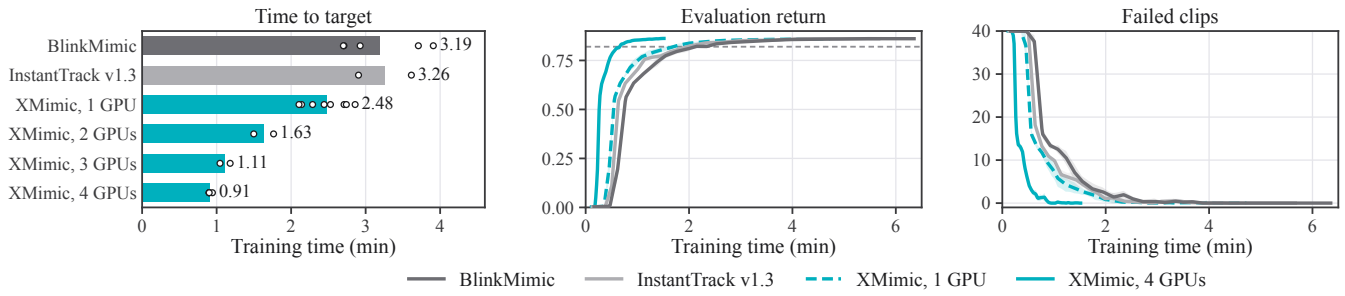


Fig. 3. Time to target (left), evaluation return (middle), and number of failed clips (right) on H100 GPUs, measured in training time without compilation. The bars show the mean and the dots the individual runs. The curves show the mean and standard deviation over the two to eight runs of each method. The dashed line marks the target return of 0.82.

A. Compiling the Training Iteration

To remove the times in which the GPU waits for Python, XMimic implements the whole iteration in JAX, including the MuJoCo Warp simulator [10], [11], and JAX compiles the iteration into a single program for the GPU. On a single GPU, we even compile 50 iterations into one program, so the GPU trains without interruption while Python only starts the next program and writes the logs. We also speed up the most expensive step, the policy update. It runs in the same 16-bit precision as in BlinkMimic, and we restructure the backpropagation through the imagined steps into fewer and larger GPU operations. For example, every computation that does not depend on the imagined state is done once before the 32 imagined steps instead of in each of them. Together, this reduces the time of a policy update by almost 40% compared to a direct translation of the BlinkMimic code.

B. Splitting the Iteration Across GPUs

On a single GPU, the three steps of an iteration run one after another, so an iteration takes as long as all three steps together. With more GPUs, XMimic runs the steps at the same time on different GPUs, similar to an assembly line (see Figure 2). An actor collects experience on the first GPU, a world-model learner fits the world model and the critic on the second GPU, and a policy learner updates the policy on the third GPU. Each of them works with the latest results of the others, which come from the previous iteration, so an iteration only takes as long as its slowest step,

$$T_{\text{single}} = T_{\text{c}} + T_{\text{w}} + T_{\text{p}}, \qquad T_{\text{split}} \approx \max(T_{\text{c}}, T_{\text{w}}, T_{\text{p}}), \quad (1)$$

where T_{c}, T_{w}, and T_{p} are the times for collecting experience, learning the world model, and updating the policy, respectively. The price is that the actor collects experience with a policy that is one iteration old, and the policy learns in a world model that is two iterations old. The fourth GPU runs the failure search, which halves the training speed of a single GPU while it runs. Such a split only pays off if no GPU waits for the CPU, so the actor runs in its own Python thread and the CPU prepares the work of every iteration in advance.

III. EXPERIMENTS

We follow the task and the evaluation of BlinkMimic, where a single policy learns to track all 40 LAFAN1 clips, about 2.5 hours of human motion retargeted to the Unitree G1. We compare XMimic to BlinkMimic and to InstantTrack v1.3, which we run unchanged with their own PyTorch code, and train every method on H100 GPUs of the same cluster. Every 100 iterations, we evaluate the policy on all clips, where a clip fails if the height or tilt of the robot deviates too far from the motion, for example when the robot falls. Following BlinkMimic, the training reaches its target once the normalized return, where 1 means perfect tracking, stays at 0.82 or higher without a single failed clip for three evaluations in a row. We report the training time until this target without the one-time compilation of the code. The only difference in the simulation is that XMimic allows fewer contact constraints per environment, which is slightly faster and only rarely limiting.

Figure 3 shows the time to target and the learning curves of all methods. On a single GPU, XMimic reaches the target in 2.48 minutes, while BlinkMimic needs 3.19 minutes. Both need about the same number of iterations, but XMimic runs every iteration faster. With more GPUs, the training time shrinks further, and four GPUs reach the target in 0.91 minutes, which is 3.5 times faster than BlinkMimic. The older policy and world model in the split iteration only cost about 10% more iterations, while the four GPUs run four times as many iterations per second. Together, the four GPUs use slightly more GPU time than BlinkMimic on its single GPU, so the split trades a little compute for a much shorter training time. All methods end at the same tracking quality, so XMimic learns an equally good policy, only sooner.

IV. CONCLUSION

We presented XMimic, which learns whole-body motion tracking for the Unitree G1 in under a minute by compiling the training iterations of BlinkMimic into single GPU programs and splitting them across up to four GPUs. With four GPUs, collecting experience in the simulator and updating the policy now take almost the entire iteration. In future work, we will therefore split the data collection across more GPUs and use the short training times to learn from larger motion datasets.

ACKNOWLEDGMENT

We acknowledge EuroHPC Joint Undertaking for awarding us access to MeluXina at LuxProvide, Luxembourg, Vega at IZUM, Slovenia, and Arrhenius at NAISS, Sweden, through the Development Access project EHPC-DEV-2026D08-135. The authors gratefully acknowledge the HPC RIVR consortium and EuroHPC JU for funding this research by providing computing resources of the HPC system Vega at the Institute of Information Science. Some of the experiments were performed on the Luxembourg national supercomputer MeluXina, and the authors gratefully acknowledge the LuxProvide teams for their expert support. Computational resources were provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), funded by the Swedish Research Council. The authors gratefully acknowledge the computing time granted by the KISSKI project. The calculations for this research were conducted with computing resources under the project kisski-merl. The authors gratefully acknowledge the computing time provided to them on the high-performance computer Lichtenberg II at the NHR Center NHR4CES at TU Darmstadt (project number p0028682). This is funded by the Federal Ministry of Research, Technology and Space, and the state governments participating on the basis of the resolutions of the GWK for national high performance computing at universities. The authors gratefully acknowledge the computing time made available to them on the high-performance computers Otus [12] and Noctua 2 [13] (project hpc-prf-smerl) at the NHR Center Paderborn Center for Parallel Computing (PC2). This center is jointly supported by the Federal Ministry of Research, Technology and Space and the state governments participating in the National High-Performance Computing (NHR) joint funding program (www.nhr-verein.de/en/our-partners). Some of the experiments ran on the hessian.AI cluster 43, and we gratefully acknowledge support from the hessian.AI Service Center (funded by the Federal Ministry of Research, Technology and Space, BMFTR, grant no. 16IS22091) and the hessian.AI Innovation Lab (funded by the Hessian Ministry for Digital Strategy and Innovation, grant no. S-DIW04/0013/003).

REFERENCES

[1] Q. Liao, T. E. Truong, X. Huang, G. Tevet, K. Sreenath, and C. K. Liu, "BeyondMimic: From motion tracking to versatile humanoid control via guided diffusion," *arXiv preprint arXiv:2508.08241*, 2025.

[2] Q. Liao, Y. Shao, J. Long, K. Zakka, R. Horowitz, P. Abbeel, and K. Sreenath, "SuperMimic: Scaling humanoid whole-body control with one differentiable world model," *arXiv preprint arXiv:2610.28480*, 2026.

[3] L. Fussell, K. Bergamin, and D. Holden, "SuperTrack: Motion tracking for physically simulated characters using supervised learning," *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia)*, vol. 40, no. 6, 2021.

[4] C. Zhang, "supermimic_vibed: An unofficial reimplementation of SuperMimic," 2026. [Online]. Available: https://github.com/zita-ch/supermimic_vibed

[5] Huy-Quang-Dao, "InstantTrack: Whole-body motion tracking for the Unitree G1 in 15 minutes on one GPU," 2026. [Online]. Available: https://github.com/Huy-Quang-Dao/instantTrack

[6] F. Chen, "BlinkMimic: Whole-body motion tracking for the Unitree G1 in minutes on one GPU," 2026. [Online]. Available: https://github.com/fuyichen0725/blinkmimic

[7] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal, "Robust motion in-betweening," *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, vol. 39, no. 4, pp. 60:1–60:12, 2020.

[8] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/jax-ml/jax

[9] I. Choi, G. Leem, and J. Won, "InstantMimic: A high performance system for learning physics-based skills in seconds," *arXiv preprint arXiv:2609.09821*, 2026.

[10] Google DeepMind and NVIDIA, "MuJoCo Warp: GPU-accelerated MuJoCo with NVIDIA Warp," 2025. [Online]. Available: https://github.com/google-deepmind/mujoco_warp

[11] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033.

[12] S. Ehtesabi, M. Hossain, T. Kenter, A. Krawinkel, H. Nitsche, L. Ostermann, C. Plessl, H. Riebler, S. Rohde, R. Schade, M. Schwarz, J. Simon, N. Winnwa, A. Wiens, and X. Wu, "Otus Supercomputer," Paderborn Center for Parallel Computing (PC2), Paderborn, Germany, PC2 Technical Report PC2TR-2025-1, Dec. 2025.

[13] C. Bauer, T. Kenter, M. Lass, L. Mazur, M. Meyer, H. Nitsche, H. Riebler, R. Schade, M. Schwarz, N. Winnwa, A. Wiens, X. Wu, C. Plessl, and J. Simon, "Noctua 2 supercomputer," *Journal of large-scale research facilities JLSRF*, vol. 9, 2024.