RoboJackson: An Imitation Learning Framework for Dance Steps

Adithi Narayan(an3308), Priyanka Rose Varghese(prv2108)

***Abstract*—This paper presents RoboJackson, a framework to teach humanoid robots to perform expressive dance movements by learning from real-world video clips. We use short dance videos from YouTube Shorts and extract 3D full-body motion using a two-stage pose estimation method combining HybrIK and NIKI. The captured motions are retargeted to a MuJoCo humanoid robot using a custom joint mapping and smoothing steps to ensure continuity. The robot is trained with reinforcement learning using the Truncated Quantile Critics (TQC) algorithm, starting from a walking model to ensure stability early in training. Our reward function encourages imitation of the dance, upright posture, balance, and exploratory motion. Through ablation studies, we show that using a pretrained model and adding curiosity improves the robot's learning speed and motion quality—particularly for more complex dances. RoboJackson highlights how lifelike movements can be learned from simple video input, paving the way for more expressive and engaging robots.**

I. Introduction

Whole-body expressiveness is a relatively new concept in humanoid robotics. Several studies [2] [4] have been conducted to bring agile, human-like movements to robots. Termed as whole body expressiveness, this can also be seen as an attempt to create more anthropomorphic robots, which do not just complete a certain task, but rather perform them in a more human-like way by expressing internal states using their bodies.

Dance is one of the fundamental ways through which humans express emotion. Enabling robots to exhibit such human-like behaviors would serve as a powerful testament to their potential for meaningful integration into human society. However, this also raises an important challenge—the lack of large-scale, diverse datasets specifically focused on humanoid or robotic dance. While AIST++ [7] provides a substantial collection of dance data, it remains limited in scope. Continuously updating existing datasets to include newer and trendier dance styles is a time-consuming and tedious process.

The best source of such videos is publicly available YouTube Shorts. Most available video data is unstructured and low in quality, often captured from a single camera without any direct motion tracking or 3D reference. Estimating accurate and smooth 3D body movements from such footage is difficult, especially over time. Even after estimating the motion, transferring it to a robot is not straightforward; human joints and robot joints often work very differently, with different joint mappings across various simulations and differences in DOFs. On top of that, the motion data can be noisy or incomplete, making it harder for the robot to learn stable movements even with training.

In order to capture motion data from human demonstrations, SMPL, a model inspired by SMPLify, captures full-body human pose and constructs a compact and expressive representation of human motion. This motion data can then be retargeted to the required robot joint data.

However due to the difference in joint mappings and DOFs, such retargeting is noisy and is not smooth. Imitation learning, a method in which a model learns in a supervised manner from expert demonstrations, combined with reinforcement learning (introduced in [10]), has shown great promise in numerous studies involving learning from human demonstrations. On a humanoid pretrained on walking, using TQC as a learning policy and a tuned reward function the humanoid was trained to learn the movements in the human demonstration. The pretrained model provides locomotion stability, while TQC fine-tunes the policy to track complex human-like motions extracted from demonstration data, allowing the robot to perform fluid and expressive movements.

In this project, we:

- Collected trending dance videos from YouTube Shorts and extracted 3D full-body motion using the SMPL model.
- Retargeted the extracted motion to a MuJoCo humanoid robot, addressing differences in joint mappings and degrees of freedom.
- Built a custom Gym environment with a reward function based on motion similarity, balance, stability and curiosity.
- Fine-tuned a humanoid pretrained on walking using the TQC reinforcement learning algorithm to learn dance movements from the retargeted motion.
- Evaluated the robot's ability to perform smooth, expressive movements that resemble the original dance demonstrations.

II. Related Works

The most recent studies in robotics focus on efforts to create anthropomorphic robots, which are robots that can convey human-like characteristics. A recent study by Apple [3] in 2025 shows their efforts in building a robot that conveys its internal states to its interactors through motion. Robots that dance demonstrate anthropomorphic qualities by mimicking human-like movement, conveying emotions and internal states through coordinated motion. Several efforts have been made in the past to develop robots that move in sync with music.

This study [2] focuses on the challenge of transferring robotic learning from simulation to real-world applications

(Sim2Real), particularly for humanoid robots performing complex, agile tasks. It uses a two-stage framework, in which the first stage motion tracking policies are pretrained in simulation using retargeted human motion data. In the second stage,the policies in the real world.

A study conducted in 2014 by Okamoto et.al [8], uses a physical humanoid robot HRP2 to dance to one of the Japanese traditional folk dances. It learns the key poses of the dance exhibited by humans and is able to perform the same series of steps even when the tempo of the dance is increased to 1.2x and 1.5 times the original music.

Another study [11], utilizes an autoregressive generative model (DanceNet) to synthesize realistic and diverse 3D dance movements by capturing the style, rhythm, and melody of music.

Robot dancing is achieved in this study [1], where optimally timed actions are executed based on beat timings and extracted musical features. It proposes three methods for robot choreography: expert-designed poses matched to music, imitation of human dance through movement mapping, and automatic generation based on musical features and optimization. This form of learning enables the robot to choreograph dances to a specific music beat. The study was conducted on the upper body of the humanoid robot RH5 Manus.

This work [9] shows how reinforcement learning (RL) can create control policies that mimic various motion clips regardless of environmental changes. It uses the flexibility offered by RL based methods to train characters that react intelligently in interactive settings.

III. Methods

Our goal was to teach a robot to dance like trending real-life performers using only publicly available YouTube shorts. This in-the-wild setting had a lot of core challenges:

- Unstructured, low-quality monocular video data
- No direct motion capture or 3D ground truth
- Need for temporally coherent 3D pose estimation
- Transfer learning of human motion to a robot with different joint constraints
- Effective RL policy learning from noisy and partial motion references

Our pipeline to tackle this was as follows:

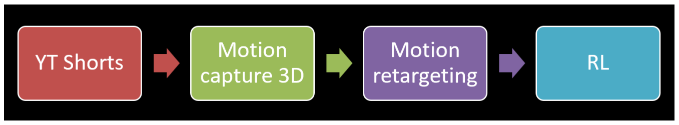


Fig. 1: Overall pipeline overview

A. Data collection

Our goal was to make the pipeline end to end automated and modular. The first step was to collect quality dance motion videos from publicly available video content. We targetted YouTube shorts instead of full length videos, as they were:

- Short [under 60sec]
- More likely to contain trendy dances
- Will mostly have one or few people

This was needed since we wanted our pipeline to be fast, efficient and did not want to deal with full length videos which might have a lot of extra/irrelevant content [music videos, tutorials, streams]. To get relevant videos we used Youtube-dl library with a simple keyword based search query corresponding to the name of the dance [Ex: APT dance]. From the fetched results, we filtered for videos less than 60sec to exclusively focus on YouTube shorts. This made sure we were more likely to get videos with one or two performers as we want to train a single agent robotic imitation setting.

We then sorted the filtered results by view count and selected the top 5 videos as candidates. Not all retrieved videos accurately contain the desired dance or had high quality content. Some didn't even have humans [Ex: One of the top 5 for APT is a roblox bot doing the dance]. To take care of this, we introduced a quality filtering mechanism based on pose stability.

Each video was processed frame by frame using Google MediaPipe for 2D pose estimation. MediaPipe was chosen because we wanted a fast, light weight and single-person focusing pose estimate. To evaulate the quality of motion in each video we defined a custom stability score that penalizes:

- Missing joints: Frames where MediaPipe failed to detect key landmark joints
- Landmark jitter: Abrupt and unnatural changes in joint positions between computed frames computed as average ℓ_2 distance between joint positions in successive frames

$$\text{Jitter}_t = \frac{1}{N}\sum_{j=1}^{N}\left\|\mathbf{L}_t^j - \mathbf{L}_{t-1}^j\right\|_2$$

where $\mathbf{L}_t$ = the set of joint landmarks at frame t

Final stability score for each video is the sum of missing joints and jitter across frames. We select the video with lowest stability score as the most reliable video for further processing

Algorithm 1 YouTube Shorts Dance Video Selection

```
Require: Dance keyword query $q$
 1: $videos \leftarrow$ YouTubeDL.search($q$)
 2: $shorts \leftarrow$ filter videos with duration $< 60$ seconds
 3: $top\_videos \leftarrow$ top 5 from $shorts$ by view count
 4: for each video $v$ in $top\_videos$ do
 5:     Extract 2D pose using MediaPipe $\rightarrow \{\mathbf{L}_t\}_{t=1}^T$
 6:     Count missing joints per frame $\rightarrow M_v$
 7:     Compute jitter: $J_v \leftarrow \sum_{t=2}^T \frac{1}{N}\sum_{j=1}^N \|\mathbf{L}_t^j - \mathbf{L}_{t-1}^j\|_2$
 8:     $score_v \leftarrow M_v + J_v$
 9: end for
10: $selected\_video \leftarrow$ video with minimum $score$
11: return $selected\_video$
```

B. 3D motion capture

Now that we have our high-quality YouTube Shorts video, the next task was to extract temporarlly consistent 3D fully body poses from it. We chose 3D pose estimation over 2D because 3D joint coordinates are more useful for retargetting and physically simulating with our humanoid agent. 2D key-points lack depth information and often suffer from perceptive distortion. 3D allows us to handle left-right ambiguity and map poses more accurately to a robot with a different structure. For this we used a two stage motion capture pipeline: Hybrik followed by NIKI to get high quality motion capture.

1) HybrIK: Hybrid Inverse Kinematics for 3D Pose Estimation.: HybrIK [5] is a state-of-the-art framework for monocular 3D human pose and shape estimation. It begins by using a convolutional neural network backbone (typically a ResNet) to regress 3D joint positions $\hat{J}_j$ relative to the pelvis (root joint) in the camera coordinate frame. Rather than directly estimating joint angles, HybrIK preserves the inverse kinematics structure of the SMPL model and solves for the pose parameters θ and shape parameters β via optimization:

$$\min_{\theta,\beta} \sum_j \|J_j^{\text{SMPL}}(\theta,\beta) - \hat{J}_j\|^2$$

Here, $J_j^{\text{SMPL}}(\theta,\beta)$ are the 3D joint positions produced by the SMPL model given pose and shape. This step ensures kinematic plausibility and enforces consistency between predicted joint locations and valid human body configurations.

However, HybrIK's predictions are often temporally noisy due to independent frame-wise estimation. It also struggles with depth ambiguities common in monocular setups. This is where NIKI is introduced.

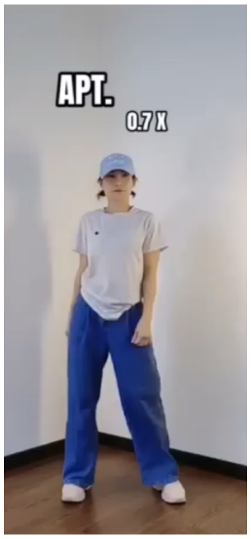
2) NIKI: Neural Inverse Kinematics with Invertible Neural Networks.: To address the limitations of HybrIK, we refine its outputs using NIKI [6], a recent method that models inverse kinematics as an invertible mapping between latent pose parameters and observable 3D joint configurations. NIKI uses an Invertible Neural Network (INN) to learn a bijective function f such that:

$$f^{-1}(x,z) = y \quad \text{and} \quad f(y) = (x,z)$$

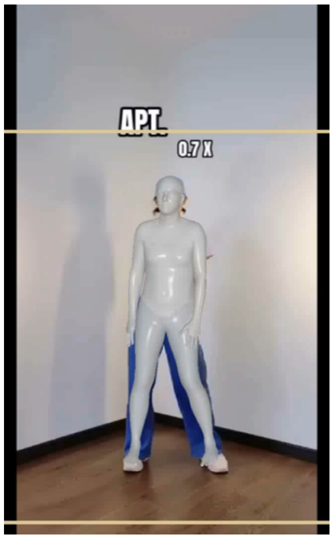
where:

- x is the observed 3D joint positions (e.g., from HybrIK),
- y is the latent pose and shape parameters (θ,β),
- z is a latent variable capturing ambiguity in the inverse mapping.

This architecture allows NIKI to generate multiple plausible poses for a given observation, resolve common left-right depth ambiguities, and enforce temporal smoothness across frames. Additionally, it improves downstream imitation learning by providing clean and consistent motion trajectories.



(a) Raw YouTube Shorts frame



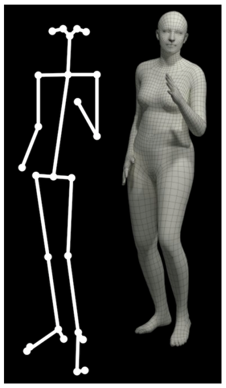
(b) 3D MoCap output (HybrIK + NIKI)

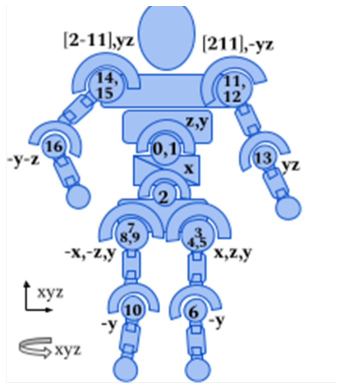
Fig. 2: End-to-end motion capture pipeline: The input is a YouTube Shorts dance frame (left), and the output is the reconstructed 3D pose (right).

3) Summary: The final output is a sequence of SMPL formatted pose parameters and 3D joint coordinates to be further processed. By combining NIKI and Hybrik we obtained motion capture data that is both plausible and temporally coherent.

C. Motion retargetting

Once we got the 3D poses in SMPL format, we had to retarget it to the simulated humanoid robot. Our simulation environment was Mujoco and the humanoid was Mujoco Gym Humanoid v5. The joints in SMPL and the humanoid are fundamentally different. The motion capture we got had 29 degrees of freedom while the humanoid had 16 degrees of freedom.


(a) 3D Motion capture



(b) Humanoid v5

Fig. 3: Motion retargeting: Have to change from SMPL motion capture data with 29 doF (left) to Humanoid-v5 with 16 doF (right)

We constructed a custom joint mapping between the SMPL joints and their corresponding Mujoco Humanoid joints. Some SMPL joints work on a single axis while others map to multiple Mujoco joints to account for full rotational doF [x,y,z axes]. Our mapping is:

SMPL Joint	**MuJoCo Joint(s)**
pelvis	root
left_hip	left_hip_x, left_hip_y, left_hip_z
right_hip	right_hip_x, right_hip_y, right_hip_z
left_knee	left_knee
right_knee	right_knee
left_ankle	left_foot
right_ankle	right_foot
left_shoulder	left_shoulder1, left_shoulder2
right_shoulder	right_shoulder1, right_shoulder2
left_elbow	left_elbow
right_elbow	right_elbow
left_wrist	left_hand
right_wrist	right_hand

TABLE I: Mapping between SMPL joints and MuJoCo humanoid joints.

This mapping ensures that core kinematic structure and motion semantics is preserved during retargetting. For joints with multiple MuJoCo components [Hips/shoulders] we applied the SMPL joint rotation vector across all corresponding axes.

1) Preprocessing and Normalization: Since there are differences in scale, we applied normalization before imitation:

- **Scaling:** We scaled the SMPL body globally to approximately match the dimensions of Mujoco Humanoid
- **Interpolation and Smoothing:** Even though we had high-quality videos and high-quality motion capture, there is still a chance that it contained missing or noisy frames. We applied interpolation and smoothing to maintain continuity. We used the `scipy.interpolate` library with linear interpolation and Gaussian smoothing to remove jitter.

Now we get our target humanoid joint data to be used to train our robot further

D. Reinforcement learning

Finally to make the robot actually learn to dance, we adopted a reinforcement learning approach. The agent learns a policy to control joint trajectories and torques so that the resulting motion mimics the human reference while maintaining physical stability.

1) Truncated Quantile Critics TQC: We used Truncated Quantile Critics, an off-policy actor-critic algorithm from sb3-contrib library. We chose TQC for the following reasons:

- Robust to reward shaping noise: Dance motion may have many small errors. TQC helps manage such variance during training
- Better exploration: Broader policy support for expressive multi-joint coordination.
- Sample efficient: Since full body motion learning is data-intensive, TQC's stability reduces training time

2) Pre-training: We initialized training with a pretrained walking model from HuggingFace. This model could walk upright by only moving its legs minimally. This motion was highly unrealistic and did not move any other part of the body except legs/ankles. This gave us critical balancing and locomotion priors significantly reducing exploration time, preventing early falling. Pretraining allowed faster convergence and better stability so we could focus on imitation specific fine-tuning.

3) Reward function: The agent's behavior is shaped by a weighted reward composed of four terms:

- **Imitation Loss:** Penalizes deviation from reference 3D joint positions, weighted per joint.
- **Uprightness Reward:** Encourages the torso to remain aligned with the vertical axis, promoting balance.
- **Center-of-Mass (COM) Reward:** Rewards stability by minimizing the horizontal distance between the robot's center of mass and pelvis.
- **Curiosity Reward:** Encourages dynamic and expressive motion by rewarding joint velocity magnitude, avoiding stagnation. Small since we want it to explore but still imitate in the end.

The total reward at timestep t is defined as:

$$R_t = -\lambda_t \cdot \underbrace{\frac{\sum_{j\in\mathcal{J}} w_j \cdot \left\| \mathbf{p}_j^{\text{mj}} - \mathbf{p}_j^{\text{ref}} \right\|^2}{\sum_{j\in\mathcal{J}} w_j}}_{\text{Imitation Loss}}$$
$$+ \underbrace{3.0 \cdot \max\left(0, \mathbf{z}^\top \mathbf{g}\right)}_{\text{Upright Reward}}$$
$$+ \underbrace{2.0 \cdot \frac{1}{1 + \|\text{COM}_{xy} - \text{Pelvis}_{xy}\|}}_{\text{COM Balance Reward}}$$
$$+ \underbrace{0.1 \cdot \sum_{i=1}^{m} \left\| \mathbf{v}_i^{\text{robot}} \right\|}_{\text{Curiosity Reward}}$$

4) Curriculum learning: We adopted a **curriculum learning** strategy to help the agent gradually improve stability and motion fidelity. The curriculum was:

- Gradually increasing the imitation reward weight λ_t: The agent begins training with a low imitation weight ($\lambda_t = 0.1$) to avoid being overwhelmed by the complex imitation task while it is still learning to balance. The imitation weight is linearly increased over the course of `imitation_schedule_steps` (set to 1000 episodes) until it reaches the maximum reward scaling factor. The update rule is:

$$\lambda_t = \lambda_{\text{max}} \cdot \min\left(1.0, \frac{\text{episode\_idx}}{\text{imitation\_schedule\_steps}}\right)$$

This ensures that the agent first maintains upright posture and locomotion before being gradually trying to accurately imitate the SMPL reference motion.

- **Fall detection and early termination:** To discourage unstable behaviour, if the agent falls [torso height below a threshold of 0.7m or torse tilt above 60 deg], it incurs a strong penalty and the episode is terminated. This discourages collapsing strategies and reinforces uprightness first in training

Our overall RL algorithm is as follows:

Algorithm 2 Complete RL algorithm

```
1:  Input: Pretrained TQC model $\pi_\theta$, reference motion $\mathcal{M}$, joint names $\mathcal{J}$
2:  Initialize environment $\mathcal{E} \leftarrow$ HumanoidImitationEnv($\mathcal{M}, \mathcal{J}$)
3:  Set curiosity weight $\beta \leftarrow 0.05$
4:  for each episode $e = 1$ to $E$ do
5:      $\mathcal{E}$.reset()    ▷ Resets frame index and updates curriculum
6:      Initialize state $s_0$
7:      while not terminated or truncated do
8:          Sample action $a_t \sim \pi_\theta(s_t)$
9:          for $k = 1$ to frame_skip do
10:             Execute $a_t$ in MuJoCo: $(s'_t, r_{\text{mjc}}, \text{done}) \leftarrow$ $\mathcal{E}$.step($a_t$)
11:             if done then break
12:
13:                 if $t < |\mathcal{M}|$ then
14:                     $r_{\text{imit}} \leftarrow$ imitation loss from SMPL $\mathcal{M}[t]$
15:                     $r_{\text{upright}} \leftarrow$ torso verticality reward
16:                     $r_{\text{com}} \leftarrow$ center-of-mass alignment reward
17:                     $r_{\text{curiosity}} \leftarrow \beta \cdot \sum_{j=1}^{m} \|\mathbf{v}_j^{\text{robot}}\|$
18:                     $r_t \leftarrow -\lambda_t \cdot r_{\text{imit}} + 3.0 \cdot r_{\text{upright}} + 2.0 \cdot r_{\text{com}} + r_{\text{curiosity}}$
19:                 else
20:                     Mark episode as terminated
21:                 end if
22:                 if is_fallen() then
23:                     $r_t \leftarrow r_t - 200$
24:                     Terminate episode
25:                 end if
26:                 Store $(s_t, a_t, r_t, s_{t+1})$ in replay buffer
27:                 Update policy $\pi_\theta$ using TQC
28:
29:                 if episode ends early ($t <$ fall_threshold) then increment early fall counter
30:
```

IV. Experiments

To evaluate our humanoid, we defined 3 movements:

1) APT1: A simple step from a dance video for APT which involves shaking the hips alone
2) APT2: A more complex dance step from APT which involves moving multiple limbs
3) Walk: A realistic walk

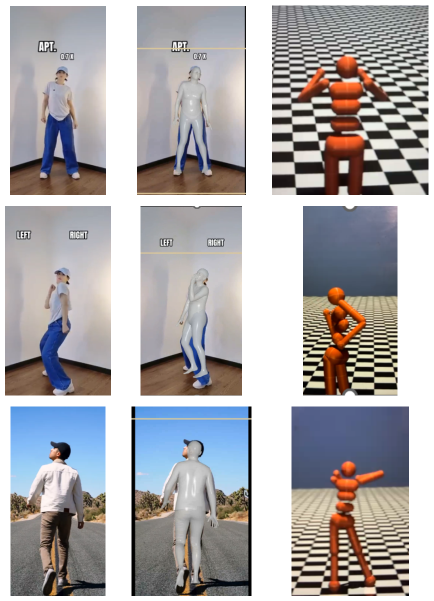


Fig. 4: Top to bottom: APT1 (simple dance), DANCE (moderately complex choreography) and WALK (realistic locomotion). Each row shows the original YouTube frame (left), the 3D MoCap result (middle), and the final robot imitation (right).

Our humanoid, trained using this pipeline, was evaluated through a series of ablation studies. We plot the *ep_reward_mean* and tthe *episode_length_mean* against the timesteps. We perform the following ablation studies:

1) Training with pretrained model
2) Training without pretrained model: We train a TQC model from scratch
3) Curiosity: The effect of curiosity with the pretrained model

Setup	Move	Ep. Reward	Ep. Length
No Pretrain	1	Low, rises at 20K	Gradual
	2	Flat, spike >30K	Very slow
	3	Slow rise	Late increase
Pretrain	1	High (>300)	High early
	2	Steady high	Improved
	3	Best (∼350+)	Consistent
Pretrain + Curiosity	1	Fast (∼400)	Moderate
	2	Fast, dips mid	Slightly unstable
	3	Lower early	Slightly worse

TABLE II: Summary across different training setups

The following results highlight the impact of each component:

A. Training Without Pretrained Initialization

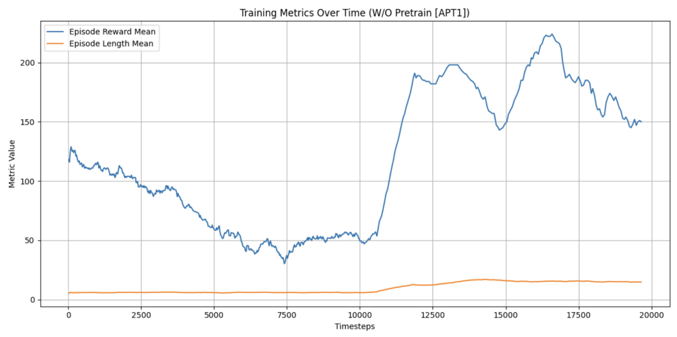


Fig. 5: Training Without Pretrained Initialization (Movement 1)

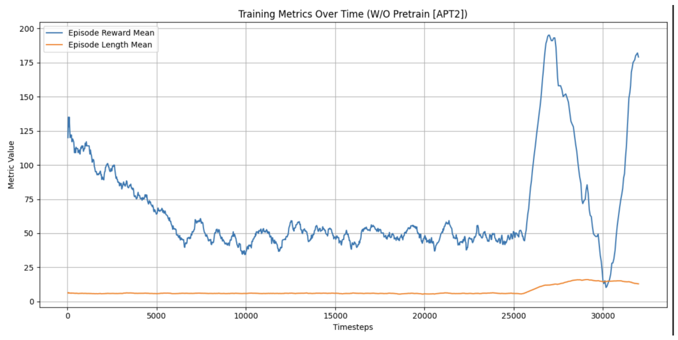


Fig. 6: Training Without Pretrained Initialization (Movement 2)

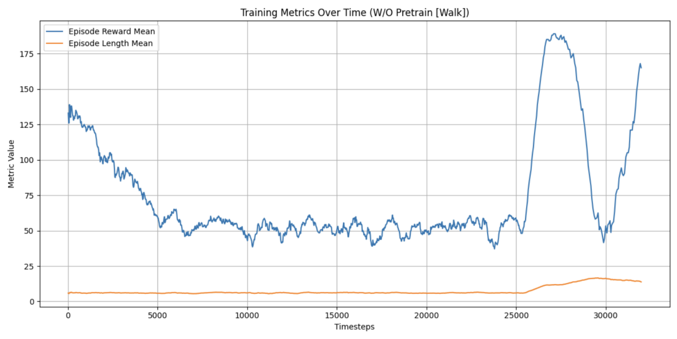


Fig. 7: Training Without Pretrained Initialization (Movement 3)

Without using a pretrained model, the training process begins with significantly lower episode rewards, especially for complex motion sequences like those in Figure. In these cases, the agent struggles to make progress in the early phases of training. For Movement 1, reward values remain low until a noticeable improvement occurs around 12,000 timesteps, eventually ramping up by 20,000. Similarly, in Movement 2 and 3, performance remains stagnant for most of the training, with reward spikes only appearing after 30,000 timesteps. These results highlight the difficulty of learning expressive motions from scratch, particularly for harder sequences (movements 2,3), and emphasize the benefit of initializing with a pretrained policy.

B. Training With Pretrained Initialization

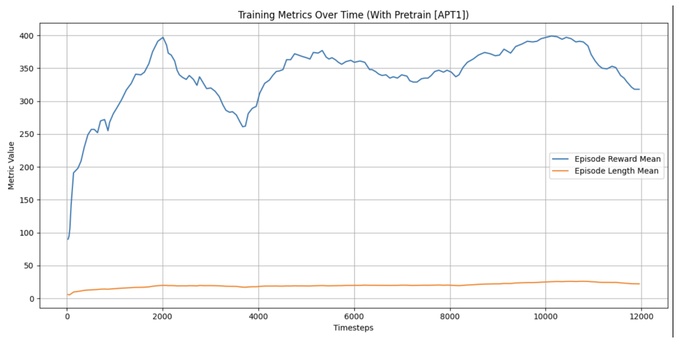


Fig. 8: Training With Pretrained Initialization (Movement 1)

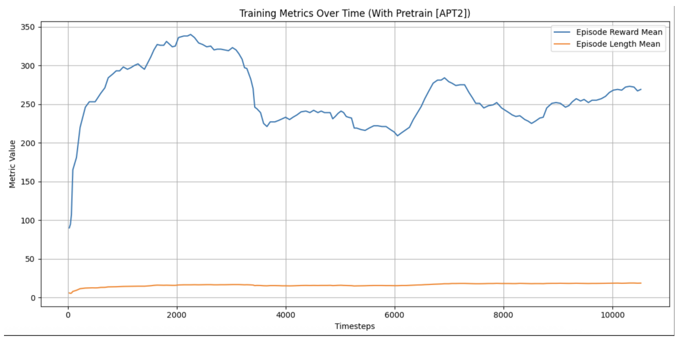


Fig. 9: Training With Pretrained Initialization (Movement 2)

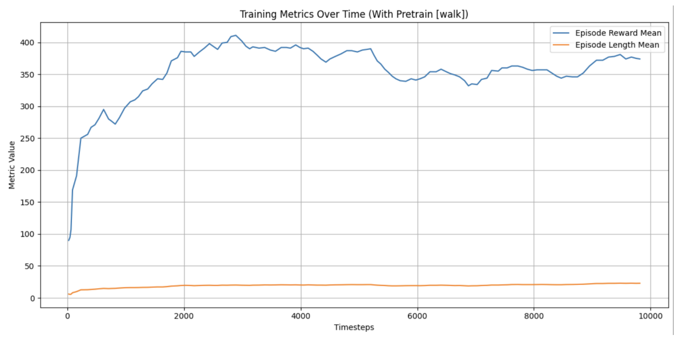


Fig. 10: Training With Pretrained Initialization (Movement 3)

With pretrained initialization, training is significantly more stable and efficient across all movements. The episode reward starts at a much higher baseline, demonstrating the benefit of initializing from a locomotion-capable model. For Movement 1, the agent quickly reaches high rewards within the first few thousand timesteps. Movements 2 and 3, which are more complex and dynamic, still show improved learning curves compared to training from scratch—reaching over 300 reward much earlier than in the non-pretrained setting. The pretrained model helps the robot maintain balance and rhythm from the start, allowing reinforcement learning to focus on refining style and motion tracking rather than learning basic locomotion.

C. *Training With Curiosity*

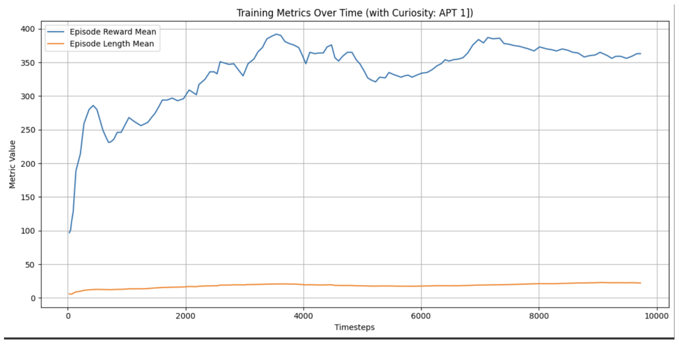


Fig. 11: Training With Curiosity (Movement 1)

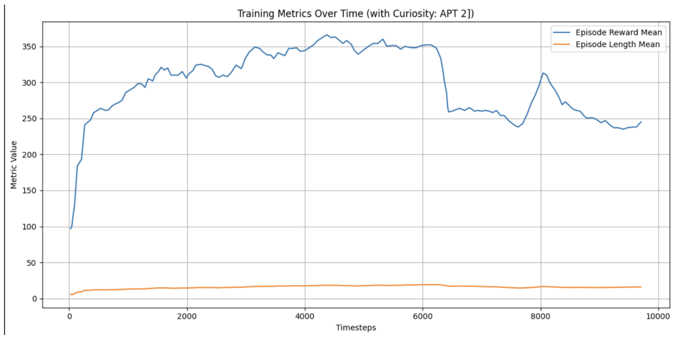


Fig. 12: Training With Curiosity (Movement 2)

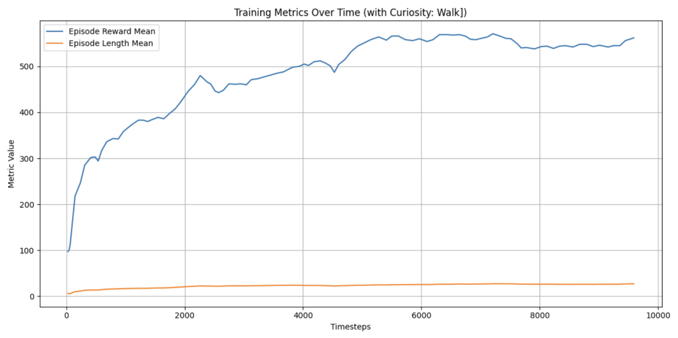


Fig. 13: Training With Curiosity (Movement 3)

Training with curiosity adds an internal reward that pushes the agent to try new and unfamiliar movements. This is helpful for complex motions like APT1 and APT2 (Figs. 10 and 11), where exploration helps the robot learn faster and reach higher rewards early on. However, for Movement 3 (Walk, Fig. 12), which is already similar to the robot's natural walking ability, curiosity can work against it. The robot starts to explore actions it doesn't need, which lowers performance during the early stages of training. This shows that while curiosity helps in learning difficult or new motions, it may slow things down when the robot already knows how to perform the task.

V. Conclusion

This project demonstrates how human motion-captured from videos using SMPL- can be effectively retargeted to a simulated MuJoCo humanoid through reinforcement learning. By designing a custom environment with joint mappings and imitation-based rewards, we trained a virtual humanoid to replicate realistic movements.

Using a pretrained TQC model, we fine-tuned the agent with rewards for imitation accuracy, balance, and upright posture. Preprocessing steps like interpolation and scaling helped smooth the motion and improve training stability. While the pipeline faces challenges such as skeletal mismatches and limited motion detail, it proves effective for transferring lifelike behaviors from real-world data.

This work lays the groundwork for building dancing or interactive humanoids that can learn directly from video - with potential extensions into music synchronization and multimodal awareness.

VI. Future Work

While RoboJackson performs well in simulation, improvements are needed. The robot sometimes exploits shortcuts, like minimizing arm movement, to increase reward without truly imitating the dance. Refining the reward function can reduce such behavior. Tracking joint usage during training may also help ensure full-body engagement.

Our work focuses on solo performances; future work could explore group dances or varied styles. Adding music-awareness, such as beat alignment, would make movements more rhythmic. Finally, transferring the system to a real robot is a crucial step toward real-world application.

References

[1] Melya Boukheddimi, Daniel Harnack, Shivesh Kumar, Rohit Kumar, Shubham Vyas, Octavio Arriaga, and Frank Kirchner. Robot dance generation with music based trajectory optimization. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3069–3076, 2022. doi: 10.1109/IROS47612.2022.9981462.

[2] Tairan He, Jiawei Gao, Wenli Xiao, Yuanhang Zhang, Zi Wang, Jiashun Wang, Zhengyi Luo, Guanqi He, Nikhil Sobanbab, Chaoyi Pan, Zeji Yi, Guannan Qu, Kris Kitani, Jessica Hodgins, Linxi ”Jim” Fan, Yuke Zhu, Changliu Liu, and Guanya Shi. Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills, 2025. URL https://arxiv.org/abs/2502.01143.

[3] Yuhan Hu, Peide Huang, Mouli Sivapurapu, and Jian Zhang. Elegnt: Expressive and functional movement design for non-anthropomorphic robot, 2025. URL https://arxiv.org/abs/2501.12493.

[4] Mazeyu Ji, Xuanbin Peng, Fangchen Liu, Jialong Li, Ge Yang, Xuxin Cheng, and Xiaolong Wang. Exbody2: Advanced expressive humanoid whole-body control, 2025. URL https://arxiv.org/abs/2412.13196.

[5] Jiefeng Li, Chao Xu, Zhicun Chen, Siyuan Bian, Lixin Yang, and Cewu Lu. Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation, 2022. URL https://arxiv.org/abs/2011.14672.

[6] Jiefeng Li, Siyuan Bian, Qi Liu, Jiasheng Tang, Fan Wang, and Cewu Lu. Niki: Neural inverse kinematics with invertible neural networks for 3d human pose and shape estimation, 2023. URL https://arxiv.org/abs/2305.08590.

[7] Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++, 2021. URL https://arxiv.org/abs/2101.08779.

[8] Takahiro Okamoto, Takaaki Shiratori, Shunsuke Kudoh, Shin'ichiro Nakaoka, and Katsushi Ikeuchi. Toward a dancing robot with listening capability: Keypose-based integration of lower-, middle-, and upper-body motions for varying music tempos. *IEEE Transactions on Robotics*, 30(3):771–778, 2014. doi: 10.1109/TRO.2014.2300212.

[9] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 37(4):1–14, July 2018. ISSN 1557-7368. doi: 10.1145/3197517.3201311. URL http://dx.doi.org/10.1145/3197517.3201311.

[10] Stefan Schaal. Learning from demonstration. In *Advances in Neural Information Processing Systems*, volume 9, pages 1040–1046. MIT Press, 1997.

[11] Wenlin Zhuang, Congyi Wang, Siyu Xia, Jinxiang Chai, and Yangang Wang. Music2dance: Dancenet for music-driven dance generation, 2020. URL https://arxiv.org/abs/2002.03761.