# PoseTriplet: Co-evolving 3D Human Pose Estimation, Imitation, and Hallucination under Self-supervision
## –*Supplementary Material*–

Kehong Gong[1,3*] Bingbing Li[2*] Jianfeng Zhang[1*] Tao Wang[1*]
Jing Huang[3] Michael Bi Mi[3] Jiashi Feng[1] Xinchao Wang[1]
[1]National University of Singapore [2]Nanyang Technological University [3]Huawei International Pte Ltd

In this document, we provide supplementary materials that cannot fit the manuscript due to page limit. Specifically, we provide details on implementation, more experimental results, and details on the pose estimator, imitator, and hallucinator, and discussion on negative social impact.

## 1. Implementation details

We implement PoseTriplet using Pytorch [5], and train PoseTriplet on a machine with a Intel Xeon Gold 6278C CPU and a Tesla T4 GPU. The whole training process takes 3 rounds for 7 days. The implementation details for each module are described as follows.

**Estimator.** In pose estimator $\mathcal{P}$, the network parameter is optimized by the Adam [3] optimizer with a learning rate of $0.0003$ and the linearly decay strategy.

**Imitator.** In pose imitator $\mathcal{I}$, the policy runs at a frequency of 30Hz. Proximal policy optimization (PPO) [9] is used to update the parameter of policy network with a learning rate of $0.00005$ and Adam optimizer. We use Mujoco [10] (open-source) as the physics simulator.

**Hallucinator.** Pose hallucinator $\mathcal{H}$ contains a generator and discriminator. The generator is trained with learning rate of $0.0001$, and the discriminator is trained with learning rate of $0.00001$. Both generator and discriminator are optimized with Adam optimizer.

## 2. Experiments

### 2.1. More qualitative results

We provide a video file "4806-supp.mp4" which includes Fig. 3-8 in video format for better visualization quality. In this video, we firstly illustrate how PoseTriplet progressively generates, refines the 3D motion data from round 1 to 3. Then we provide more self-generated examples from both imitator and hallucinator. After that we provide pose estimation results similar to Fig. 3-6 in source dataset (*i.e.*, H36M), cross dataset (*i.e.*, 3DHP and 3DPW), and videos from in-the-wild scenario (*i.e.*, self-collected from TikTok [2] and Youtube). Finally we provide more results for imitator in Fig. 7 and hallucinator in Fig. 8. Note that all of these results are achieved with only monocular 2D pose or video, *without* any 3D data or multi-view setting. This demonstrates the robustness of our PoseTriplet to challenging scenarios, again even without the necessity of acquiring 3D data.

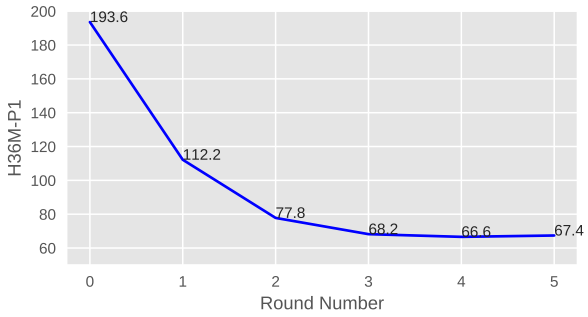


Figure 1. Results w.r.t. P1 in H36M with GT2D about Round Number.

### 2.2. More ablation studies

#### 2.2.1 Ablation on extra round of co-evolving

We here explore more rounds of co-evolving. As shown in Fig. 1, the performance saturated after round 4, implying the framework has reached the ceiling of its capacity.

#### 2.2.2 Ablations on co-evolving components

To verify the effectiveness of each component in this PoseTriplet framework, we here ablate the performance of the estimator without using imitator or hallucinator or both.

*Equal contribution.
Email: gongkehong@u.nus.edu, l.libingbing@gmail.com, zhangjianfeng@u.nus.edu, twangnh@gmail.com, jing.huang1408@gmail.com, michaelbimi@yahoo.com, jshfeng@gmail.com, xinchao@nus.edu.sg

The result in H36M with GT2D w.r.t. P2 are E: 115.8, E+H: 80.3, E+I: 57.6, E+I+H: 45.1, with E, I, H being estimator, imitator, hallucinator. This demonstrates that involving either I or H into the framework produces better results. Combining them jointly improves the performance by a large margin.

#### 2.2.3 Ablation on camera pose generator.

We use two camera projection methods (random- and GAN-based camera projection). As shown in Table 1 (Random and GAN denote random and GAN based camera projection), either method achieves a good performance, combining them together further improves the results.

| Camera projection | GT | Det |
|---|---|---|
| Random | 74.1 | 80.5 |
| GAN | 75.6 | 82.9 |
| Random+GAN | 68.2 | 78.0 |

Table 1. Ablation on camera projection in H36M (P1).

#### 2.2.4 Ablation on the physics-based pose metrics.

While results of the estimator suffer from physically implausible artifacts (*e.g.*, foot skating (FS) and ground penetration (GP)), the imitator helps resolve these issues as shown in Table 2. This ensures the plausibility of training set to sever as training data for pose estimator during co-evolving.

| Method | Train | | Test | |
|---|---|---|---|---|
| | FS | GP | FS | GP |
| w/o I | 7.1 | 2.4 | 6.5 | 3.8 |
| w/ I | 0.7 | 0.9 | 0.9 | 1.5 |

Table 2. Results on H36M train and test sets w.r.t. physics-based metrics.

## 3. Methodology

PoseTriplet is the first attempt to train three difference pose related tasks jointly in an self-supervised manner. Therefore we choose those modules based on solid and simple strategy, and they are flexible to replace by other modules with similar function. In the following section, we will elaborate the more details for the pose estimator, imitator, and hallucinator.

### 3.1. Pose estimator

For the pose estimator, we adopt the 1D convolution based VideoPose [6]. Fig. 2 illustrates the architecture of pose estimator. It contains two branches: root relative pose estimation and root trajectory estimation. Given the input 2D pose sequence $\boldsymbol{x}_{1:T}$, these two branches can predict the root relative pose and root trajectory, which are then combined as the output 3D pose sequence $\boldsymbol{X}_{1:T}$, and served as reference motion for imitator.

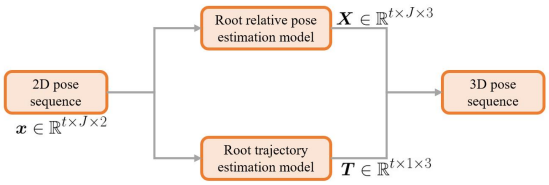


Figure 2. **Pose estimator overview.**

### 3.2. Pose imitator

We here provide more detailed information on pose imitator $\mathcal{I}$. As shown in Fig. 3, it contains reference motion, state, policy, action, and simulation environment. Note that pose representation under imitation learning is based on joint axis angle $\boldsymbol{q}$, which is convertible with joint position $\boldsymbol{X}$ through forward/inverse kinematics [4]. Unless otherwise specified, the pose in this module is represented by joint axis angle $\boldsymbol{q}$ (*i.e.*, reference motion $\boldsymbol{q}_{ref}$, velocity $\boldsymbol{q}'$, *etc.*).

**State** As shown in Fig. 3, the state includes current pose $\boldsymbol{q}_t$, current velocity $\dot{\boldsymbol{q}}_t$ from the simulation environment, target pose $\widetilde{\boldsymbol{q}}_{t+1}$ from reference motion, and an extra encoded feature $\phi$ encoded by a temporal convolution network with a receptive field of 18, which fuses the past and future reference motion information.

**Action** involves two types of forces: internal force $\boldsymbol{\tau}_t$ and external force $\boldsymbol{\eta}_t$ as shown in Fig. 3. The internal force is applied by actuator on the non-root joints (*e.g.*, elbow, knee). Following previous work [8, 12, 13], we use PD (proportional–derivative) control for internal force control. The internal force is formulated as:

$$\boldsymbol{\tau}_t = \boldsymbol{k}_p(\boldsymbol{u}_t^{nr} - \boldsymbol{q}_t^{nr}) - \boldsymbol{k}_d\dot{\boldsymbol{q}}_t^{nr}, \tag{1}$$

Where $\boldsymbol{k}_p, \boldsymbol{k}_d$ are the PD control parameters, $\boldsymbol{u}_t^{nr}$ is the target angle for PD controller, $\boldsymbol{q}_t^{nr}, \dot{\boldsymbol{q}}_t^{nr}$ is current joint pose and joint velocity, $\star^{nr}$ denotes non-root joint for non-root force $\boldsymbol{\tau}_t$ computation. The internal force $\boldsymbol{\tau}_t$, adjusted through the PD control parameters $\boldsymbol{k}_p, \boldsymbol{k}_d$ and target angle $\boldsymbol{u}_t^{nr}$ regressed by the policy network, drives the agent to target joint pose $\widetilde{\boldsymbol{q}}_{t+1}$ in desired time. The external force $\boldsymbol{\eta}_t$ is a virtual force applied on root joint (*i.e.*, hip) [12] for extra interaction (*e.g.*, sitting on the chair) and is regressed by the policy network.

**Rewards** measure the motion differences between the agent and reference motion. Following the work [7, 12], the re-

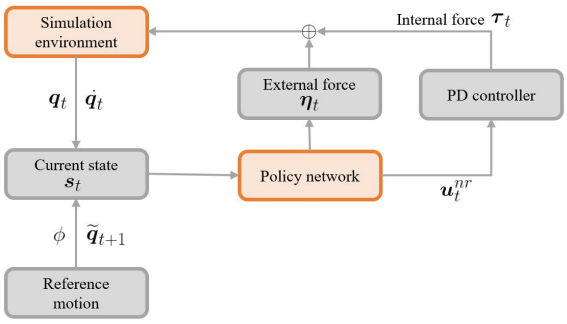


Figure 3. **Pose imitator overview.**

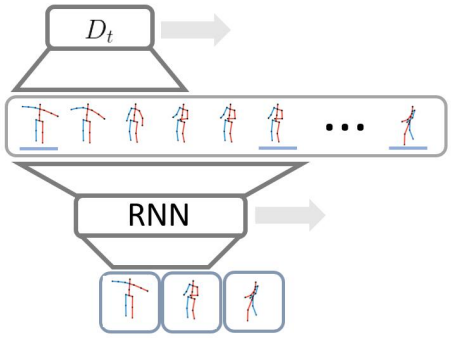


Figure 4. **Pose hallucinator overview.**

ward $\boldsymbol{r}_t$ here is formulated as:

$$\boldsymbol{r}_t = \sum_{i=1}^{n} \boldsymbol{w}^i e^{-\boldsymbol{k}_i \| \varphi_t^i \otimes \tilde{\varphi}_t^i \|}, \tag{2}$$

where $\boldsymbol{w}^i, \boldsymbol{k}_i, \varphi_t^i, \tilde{\varphi}_t^i$ are the weight factor, scale factor, motion characteristics, and reference motion characteristics, respectively. The operation $\otimes$ compute the distance between the agent motion property $\varphi_t^i$ and reference motion property $\tilde{\varphi}_t^i$. Such differences capture pose related (pose, velocity), root related (root height, root velocity) and body-end factors (position, velocity in end-effector *i.e.*, feet, hand and head). Besides, a regulation loss on virtual force is applied to avoid unnecessary external force [12], which is formatted as:

$$\boldsymbol{r}_t^{\boldsymbol{\eta}} = e^{-\boldsymbol{k}_{\boldsymbol{\eta}} \| \boldsymbol{\eta}_t \|}, \tag{3}$$

where $\boldsymbol{r}_t^{\boldsymbol{\eta}}, \boldsymbol{k}_{\boldsymbol{\eta}}, \boldsymbol{\eta}_t$ are the virtual force rewards, scale factor, virtual force, respectively. As we find that it is hard for the agent to move with the above setting due to the noisy reference motion, we further introduce a feet relative position (*i.e.*, a vector from left foot point to right foot) into the motion characteristics to enhance the feet motion. Note that all these motion characteristics are measured in local heading coordinate system, following previous works [11, 12].

**Initial Condition** For the initial condition, instead of using the starting pose [11, 12], we set the initial pose as T pose. Since the pose signal in our PoseTriplet contains more noises compared to conventional RL tasks, it may yield the initialization crash; such an initial condition, in practice, helps us to avoid simulation crush.

**Termination Condition** For the termination condition, we set it as when head height is 0.3 m below the reference head height or episode (*i.e.*, reference motion) end.

### 3.3. Pose hallucinator

For the pose hallucinator, we adopt the motion interpolation based approach [1] as it can generate arbitrary length sequence with continuously input key frames. Fig 4 illustrates the architecture of pose hallucinator. It contains a conditional RNN generator, and a temporal discriminator. During training stage, The RNN generator, condition on the sampled key frames, recover the masked intermediate frames. The temporal discriminator sliding on the recovered pose sequence, provides guidance to encourage the temporal motion plausibility. Another reconstruction loss is applied to measure the difference between recovered pose sequence and original one. During the inference stage, random sampled key frames are used as input, and generate unseen motion though the conditional RNN generator, which is then used as diversified reference motion for the imitator.

## 4. Negative social impact

Our method can be applied to lots of 3D pose estimation related applications including action recognition and human tracking, etc, but may involves user privacy issues.

## References

[1] Félix G Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. Robust motion in-betweening. In *ACM Trans. on Graphics*, 2020. 3

[2] Yasamin Jafarian and Hyun Soo Park. Learning high fidelity depths of dressed humans by watching social media dance videos. In *CVPR*, 2021. 1

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICCV*, 2015. 1

[4] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Trans. on Graphics*, 36(4):44, 2017. 2

[5] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPSw*, 2017. 1

[6] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with tem-

poral convolutions and semi-supervised training. In *CVPR*, 2019. 2

[7] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. In *ACM Trans. on Graphics*, 2018. 2

[8] Xue Bin Peng and Michiel van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In *ACM*, 2017. 2

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, 2017. 1

[10] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 1

[11] Ye Yuan and Kris Kitani. Ego-pose estimation and forecasting as real-time pd control. In *ICCV*, 2019. 3

[12] Ye Yuan and Kris Kitani. Residual force control for agile human behavior imitation and extended motion synthesis. In *NeurIPS*, 2020. 2, 3

[13] Ye Yuan, Shih-En Wei, Tomas Simon, Kris Kitani, and Jason Saragih. Simpoe: Simulated character control for 3d human pose estimation. *arXiv*, 2021. 2