Hierarchical Model-based Imitation Learning for In-between Motion Synthesis

1st Yuqi Yang
Institution of Artificial Intelligence and Robotics, Xi'an Jiaotong University
Xi'an, China
yangyuqi@stu.xjtu.edu.cn

2nd Yuehu Liu
Institution of Artificial Intelligence and Robotics, Xi'an Jiaotong University
Xi'an, China
liuyh@mail.xjtu.edu.cn

3rd Chi Zhang
Institution of Artificial Intelligence and Robotics, Xi'an Jiaotong University
Xi'an, China
chizhang@xjtu.edu.cn

***Abstract*—Motion in-between problem is the key to solve the issue of long-term motion synthesis, which are widely applied in animation production, virtual reality, video games and film industry. Motion in-between can be defined as a process generating transitions between previous motion and future motion. Previous researchers focus on motion sequence generation, which omits the interaction with environment and can cause foot-sliding problem. In this work we present a novel, generic system based on imitation learning for motion in-between problem. This hierarchical system contains two parts: first part is motion primitives, which utilize reference motion to learn general knowledge of human motion; second part is policy controller, which uses multiple primitives to generate natural human motion. The primitives are controlled by the combine weights generated by policy controller. Experiment shows that our method can effectively improve motion quality and reduce the amount of reference motion data.**

***Index Terms*—Motion In-between, Hierarchical Model, Imitation Learning.**

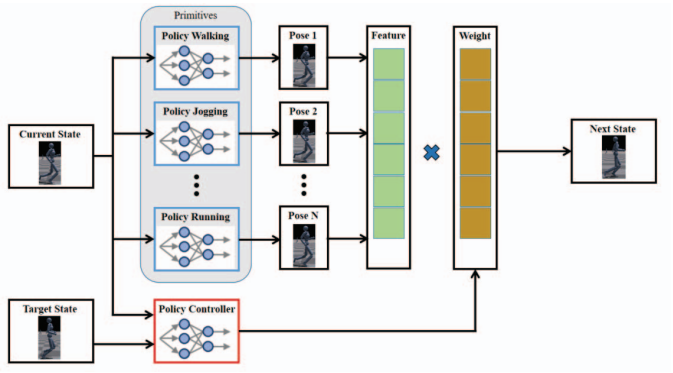

Fig. 1. System Overview. Primitives are pre-trained as certain style motion generator. Policy controller combines primitives to output the motion required.

I. Introduction

Realistic and nature long-term human motion synthesis is an important but challenging problem in many fields, including real-time character motion synthesis in VR, animating characters in films or animations and life-like motion synthesis in video games or other applications. Long-term human motion is considered complex and nondeterministic. One way to simplify the long-term motion synthesis is called "motion in-between". It generates transition between two short-term motion key-frames to make them concatenate in a natural and coherent way.

Traditional animation techniques [1] focus on manipulating key-frames between two motion clips. Animators have to draw motion frames between key-frames to make the character in a suitable position. Some improvements, including Motion Capture(MOCAP) technologies [2], is more expensive than manually drawn animation methods. Recently, learning-based methods emerge, which can generate smooth transition frames from sparse frames by learning motion capture data, including RNN-based models [3], [4], LSTM-based models [5], and BERT-based models [6].

However, methods above have a same problem: they have no interaction with the environment. It can cause foot-sliding, which makes the motion unnatural. In order to solve the problem, we introduce a hierarchical framework based on Generative Adversarial Imitation Learning(GAIL) [7]. It contains a high-level policy controller and a set of low-level motion primitives. The high-level controller extracts the information from the environment and then outputs actions. In practical terms, it generate a group of weights which combine the primitives output actions. When target key-frame is determined, the controller will be rewarded if it generates the motions leading to target pose. Motion Primitives is a set of reusable policies based on GAIL. Primitives use GAIL's framework to learning general knowledge of one kind of human motion, which can generate nature motion and exploit the learning ability of network. The contributions of this work are as follows:

1) We introduce a method to learn in-between motion synthesis policies, which enable the model to generate smooth and coherent transition motion between key-frames with less reference motion data.
2) A framework of applying imitation learning in hierarchical model is proposed, and can be utilized in other down-streaming task.
3) We provide a comprehensive analysis, which explain the outperformance of our system over other state-of-the-art approaches, including representative RNN-based and Transformer-based methods.

2023 China Automation Congress (CAC) | 979-8-3503-0375-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/CAC59555.2023.10451613

II. Related Work

A. In-between motion synthesis

In-between motion synthesis is originated from motion prediction problem. The generated motion is constrained on past and future key-frames. The most common solution is interpolating key-frames based on splines, which enables animators to control the transition precisely but it is often time-consuming. Furthermore, key-frame interpolation cannot create coherent transitions with much details. Some of the method ,including definite the problem as a Maximum Aposterior Optimization problem(MAP) [8], Gaussian Process [9], and Markov models [10] achieve successful results. As for deep learning methods, Harvey et al [11] proposed Recurrent Transition Network(RTN) to generate transitions based on target frame, previous frame and offset of the target. Some recent work utilize transformer-based models to predict the whole motion sequence [6]. However, these methods can cause problems like foot sliding, due to no or less interaction with environment.

B. Imitation learning

The challenge of generating natural motion has inspired the data-driven physics-based method developing. These method uses motion data to improve the quality of character motion. Reference motions are set to be the imitation objective of the system. This kind of method is combined with adversarial algorithm as known as GAIL [7]. Despite all this, GAIL can be unstable and the motion quality is always unsatisfactory. Peng et al. [12]produce more realistic motion in different style by regularizing the discriminator. However, their methods aim to generate motions in one style. Therefore, it is hard to be applied on motion in-between problem, for the post frame and target frame can be different motion style.

III. Methodology

A. Problem setting

Motion in-between can be set as a question that the both ends of motion sequence are fixed. The input of model should be post state and target pose. The output of model should be a sequence of character action, which can generate a sequence pose.

Given previous state $\bar{s_0}$, target pose $\bar{s_t}$, the network should generate a set of actions $a_0, a_1, ..., a_{t-1}$, which respectively guide the character transiting from s_0 to s_1, from s_1 to s_2 , and finally reaching target s_t. All the states from s_1 to s_t are considered as in-between motion. Model formulation can be defined as following:

$$(s_1, s_2 ... s_t) = Network\,(\bar{s_0}, \bar{s_t}) \tag{1}$$

B. Overview

Due to the learning limitation of one single policy network, we choose multiple policy networks for learning different styles of motion. To integrate the outputs of that networks, we introduce a hierarchical model with a policy controller to refine the outputs. All of the networks can gain information from the environment.

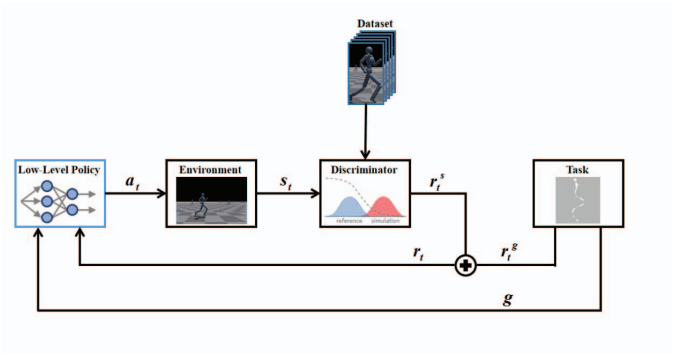

Fig. 2. Schematic overview of primitives. Given the certain motion style datasets, the system trains a discriminator which provide style reward r_t^s for policy network. Task reward r_t^g depends on specific tasks. Both rewards are combined and used to train the policy which can drive the character to generative the motion satisfactory to the task.

The system overview is shown as Fig.1. It contains a policy controller and a set of primitives. Unlike common imitation learning methods which uses single policy network, we introduce hierarchical model. The low-level primitives are pre-trained policy network. The high-level controller combines motions given by primitives to adapt to the environment and target pose. The primitives take the states and generate actions, while the controller takes the states and generate a set of weight to combine the actions. Finally, the system can generate a sequence of action to match the target pose.

C. Low-lever primitive

Figure 2 shows the training process of primitives. In the initial GAIL framework, character need to know the actions at each moment in the training data. However, when training data is shown in motion clips, the actions taken by the character is unknown, while the state at each moment is known. Therefore, the "state-action" pair, which is the objective of GAIL, can be transformed as state transition pair as following:

$$\arg\min_{\theta} -E_{d^M(s,s')}[log(D(s,s'))] - E_{d^\pi(s,s')}[1 - log(D(s,s'))] \tag{2}$$

$d^M(s,s')$ and $d^\pi(s,s')$ denotes observing the state transition from s to s' probability in datasets M and policy π . The formula above adopts sigmoid cross entropy loss function. However, in the training process, sigmoid loss function can be unstable. Here we introduce least square objective function LSGAN [13], as follows:

$$\arg\min_{\theta} E_{d^M(s,s')}[(D(s,s')-1)^2] - E_{d^\pi(s,s')}[(D(s,s')+1)^2] \tag{3}$$

The discriminator trains by solving the least squares regression problem, determining that the score of the real datasets sample is 1, and the score of the sample generated by the

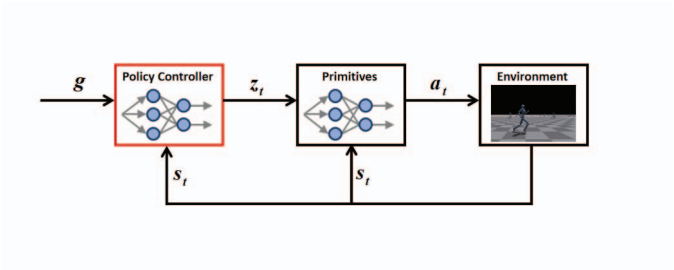

Fig. 3. Schematic overview of controller. When training, current state s_t will be sent in both controller and primitives. Controller will generate a set of weights Z_t due to target pose g and s_t, which will be combined with actions generated by primitives to produce action a_t for character.

policy network is -1. The setting of the state reward function is:

$$r(s_t, s_{t+1}) = max[0, 1 - 0.25(D(s_t, s_{t+1}) - 1)^2] \quad (4)$$

It is a common practice for Reinforcement learning network training to control the training reward between the binary values of (0,1), which can minimize the Pearson χ^2 divergency [13] between datasets M and policy π .

The policy network contains 3 hidden layers, where the first and second layers both contain 1024 units and the third layer contain 512 units. All of the layers are followed by linear output units. The value function is calculated by another similar network, with only a single linear output unit.

D. High-level controller

Figure 3 shows the training process of policy controller. After pre-training, the high-level policy network will be used to guide the motion synthesis of low-level primitives. It takes the current state s_t and target pose g as input and outputs a set of weights Z_t . The rewards for high-level policy are determined by the combination of style imitation r_t^s , time r_t^τ , and speed constraints r_t^v of task rewards:

$$\begin{cases} r_t = \omega_s * r_t^s + \omega_\tau * r_t^\tau + \omega_v * r_t^v \\ r_t^s = exp(\|\bar{x}_t - x_t\|^2) \\ r_t^\tau = exp(-(T-t) * \|x^* - x_t\|^2) \\ r_t^v = exp(\left\|v_t^{root} - v_{t-1}^{root}\right\|^2) \end{cases} \quad (5)$$

ω_s , ω_τ and ω_v are parameters. After pre-training, discriminator parameters of primitives will not be updated in high-level policy training, so there is no need to use any action data clips in high-level policy training. Previous studies [14] have found that training a discriminator with a fixed parameter may lead to unnatural actions that utilize the discriminator's feature. However, low-level policy networks that grasp the fundamental feature of motion can greatly limit the behavior of characters and to some extent reduce this impact.

The high-level policy contains 2 hidden layers with 1024 units and 512 units, both followed by a linear output unit. Output from the policy controller determine a set of weights Z_t, which will be normalized and sent to primitives policy network for next.

IV. Experiment

A. Setup

The environments are simulated by using Isaac Gym [15], a GPU-based physics simulator. The simulation operates at 120Hz, while the policy frequency is at 30Hz. All neural networks are implemented by PyTorch [16]. Reference motion clips are from CMU MOCAP motion datasets [17], including single motion style clips for training and multi motion style clips for testing.

B. Evaluation metric

The model is evaluated by the L2P, L2Q and NPSS metrics. The L2P defines the average L2 distances of the positions between the ground truth motion sequence and the generated sequence. The L2Q defines the average L2 distances of the global quaternions correspondingly. NPSS, which is based on angular frequency comparisons with the ground truth is also calculated as reference. The metrics of L2 distances are defined as following:

$$L2P = \frac{1}{NT} \sum_{n=0}^{N-1} \sum_{t=0}^{T-1} \left\| \bar{x_n^t} - x_n^t \right\| \quad (6)$$

$$L2Q = \frac{1}{NT} \sum_{n=0}^{N-1} \sum_{t=0}^{T-1} \left\| \bar{q_n^t} - q_n^t \right\| \quad (7)$$

q_n^t refers to the rotation quaternion of the generated motion sequence n at time t, while x_n^t refers to the position of generated motion. T refers to the length of each sequence and N refers to the number of sequences.

C. Low-level primitive motion synthesis

TABLE I
Performances of different primitives

Motion Style	Iteration($*10^4$)	Normalized Return
Running	5	0.93
Walking	5	0.92
Jogging	5	0.90

In this section, we demonstrate the high quality of the motion generated by low-level policy network. Figure 4 shows the learning curves of 3 primitives. Table 1 records the performances of the low-level policy network. It shows that the low-level primitives can generate high quality motions with high effectiveness. Different styles of motion, like running, walking and jogging all can be generated smoothly, which will be used to generate motion for human locomotion in motion in-between problem.

D. High-level controller

We choose two baselines as comparison. The first is keyframe linear interpolation, which linearly interpolates the root position, and spherically interpolates the quaternion of all the joints by utilizing Slerp. The second baseline is using

TABLE II
PERFORMANCES OF ALL MODELS ON MOTION IN-BETWEEN PROBLEM. ALL MODELS ARE TRAINED WITH TRANSITION IN 5,15,30 FRAMES.

	L2P			L2Q			NPSS		
length(frames)	5	10	15	5	10	15	5	10	15
Interpolation	0.37	1.25	2.32	0.22	0.62	0.98	0.0020	0.0381	0.2011
FCN	0.25	0.69	1.63	0.20	0.44	0.77	0.0030	0.0280	0.1524
ERD-QV [5]	0.23	0.65	1.28	0.17	0.42	0.69	0.0020	0.0258	0.1328
Δ-Interp [18]	0.13	0.47	1.00	0.11	0.32	0.57	0.0014	0.0217	0.1217
Ours	**0.15**	**0.64**	**0.98**	**0.15**	**0.33**	**0.54**	**0.0015**	**0.0208**	**0.1324**

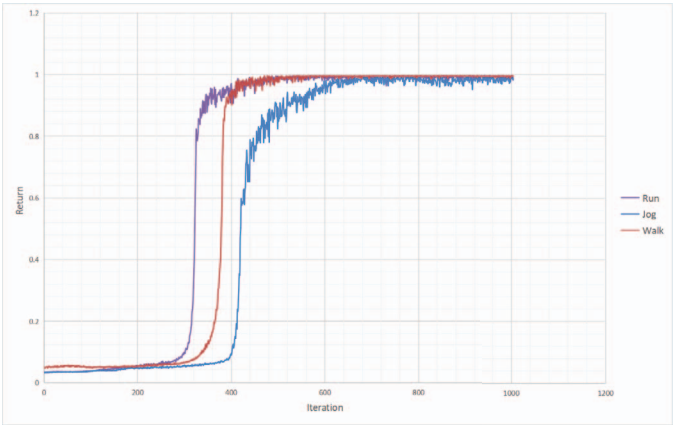

Fig. 4. Learning curves of 3 primitives. Purple represents the learning curve of primitive of running motion, while blue represents jogging and red represents walking.

Fully-Convolutional Network(FCN). Besides, we compare our work with ERD-QV [5] and Δ-Interp [18]. The ERD-QV is an Encoder-Recurrent-Decoder network. The Δ-Interp is a concurrent work using transformer encoder-decoder architecture. The result of experiment is shown in table 2.

E. Analysis

The two baseline methods show poor performance than other methods, though FCN method can make preferable performance on shorter length of transition motion synthesis. However, when the transition length grows, the motion generated by FCN starts losing details. Compared with delta and ERD-QV, the sequence consistency and velocity consistency of the transition motion generated by our method performs better. ERD-QV generates motion mostly based on the previous motions, which can cause sudden changes due to lack of global planning. Δ-Interp predicts the motion through Slerp between keyframes, which can cause fake motions. Our method can avoid such condition due to the directly generating transition process. As shown in table 2, our method show better performance on different lengths of transition motion systhesis.

V. CONCLUSION

This paper presents a novel framework that generates in-between motion by using hierarchial model-based imitation learning. It provides better results on transition motion synthesis than existing RNN and Transformer-based methods. Our method has excellent robustness in generating transitions with variable lengths. The model generates coherent motion based on less motion data. Our network can be trained at a high speed by utilizing the parallel processing of GPU. In conclusion, compared with state-of-the-art approaches, our method make a great progress in in-between motion synthesis task.

REFERENCES

[1] Ciccone L, Öztireli C, Sumner R W. Tangent-space optimization for interactive animation control[J]. ACM Transactions on Graphics (TOG), 2019, 38(4): 1-10.

[2] Holden D. Robust solving of optical motion capture data by denoising[J]. ACM Transactions on Graphics (TOG), 2018, 37(4): 1-12.Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999).

[3] Harvey F G, Pal C. Recurrent transition networks for character locomotion[M]//SIGGRAPH Asia 2018 Technical Briefs. 2018: 1-4.

[4] Holden D, Saito J, Komura T. A deep learning framework for character motion synthesis and editing[J]. ACM Transactions on Graphics (TOG), 2016, 35(4): 1-11.

[5] Harvey F G, Yurick M, Nowrouzezahrai D, et al. Robust motion in-betweening[J]. ACM Transactions on Graphics (TOG), 2020, 39(4): 60: 1-60: 12.

[6] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[7] Ho J, Ermon S. Generative adversarial imitation learning[J]. Advances in neural information processing systems, 2016, 29.

[8] Chai J, Hodgins J K. Constraint-based motion optimization using a statistical dynamic model[M]//ACM SIGGRAPH 2007 papers. 2007: 8-es.

[9] Wang J M, Fleet D J, Hertzmann A. Gaussian process dynamical models for human motion[J]. IEEE transactions on pattern analysis and machine intelligence, 2007, 30(2): 283-298.

[10] Lehrmann A M, Gehler P V, Nowozin S. Efficient nonlinear markov models for human motion[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014: 1314-1321.

[11] Harvey F G, Pal C. Recurrent transition networks for character locomotion[M]//SIGGRAPH Asia 2018 Technical Briefs. 2018: 1-4.

[12] Peng X B, Abbeel P, Levine S, et al. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills[J]. ACM Transactions On Graphics (TOG), 2018, 37(4): 1-14.

[13] Mao X, Li Q, Xie H, et al. Least squares generative adversarial networks[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2794-2802.

[14] Peng X B, Ma Z, Abbeel P, et al. Amp: Adversarial motion priors for stylized physics-based character control[J]. ACM Transactions on Graphics (TOG), 2021, 40(4): 1-20.

[15] Makoviychuk V, Wawrzyniak L, Guo Y, et al. Isaac gym: High performance gpu-based physics simulation for robot learning[J]. arXiv preprint arXiv:2108.10470, 2021.

[16] Paszke A, Gross S, Massa F, et al. Pytorch: An imperative style, high-performance deep learning library[J]. Advances in neural information processing systems, 2019, 32.

[17] CMU. [n.d.]. CMU Graphics Lab Motion Capture Database., http://mocap.cs.cmu.edu/.
[18] Oreshkin B N, Valkanas A, Harvey F G, et al. Motion Inbetweening via Deep Δ-Interpolator[J]. arXiv preprint arXiv:2201.06701, 2022.