Reinforcement Learning of Motion from Videos

Student: Xuefei Li and Advisor: Stephen J. Guy

University of Minnesota, twin-cities

1 Introduction

Physics-based simulation is mainly about passive phenomena of objects, however, more engaging animated characters are needed in virtual world or games. Modeling motions of humans and animals remains a challenging task. Manually made controller[14] by professional artists could exhibit stylized, engaging and endearing behavior, but it doesn't have generality, dynamic surroundings and physical interactions can create scenarios where premade motions are no longer suitable: a character might be pushed by a large force, or placed in a terrain with obstacles. Another drawback is that it requiring considerable effort to construct, the range of motions is limited to the space of all possible reactions. In order to produce more motions for simulated characters, adopting motions from real-world could be a solution. Motions can be obtained from Motion Capture (Mocap) data. Mocap system uses tracking cameras or non-optical approaches to measure inertia or mechanical motions. Mocap data have many advantages over traditional computer animation of a 3D model, for example, low latency, complex movement and realistic physical interactions. While mocap requires specific hardware and special software programs to obtain and process the data, it will be expensive to obtain a large dataset. In this project, I tried to refer to daily videos, which can be easily obtained by cellphones, as an abundant and flexible source of motions to learn from. These in-the-wild videos with less accuracy and stability are difficult to infer.

Our approach can be divided into two major steps. The first step is to estimate pose from the video, using temporal network architecture that produces kinematically plausible motion sequences, followed by an adversarial learning framework that leverages an existing large-scale motion capture dataset (AMASS) to discriminate between real human motions and the estimated pose. The second step is to transfer the motion to the character. Reinforcement learning (RL) is able to train a controller to perform various tasks through trial-and-error. The training process directly rewards the learned controller for producing motions that resemble reference animation data, while also achieving additional task objectives.

2 Related work

2.1 Monocular Human Pose Estimation

2D Keypoints detection from images is becoming mature, there are several tools to achieve the goal[20] [1] [5]. OpenPose[3] represents a real-time multi-person

system that jointly detects 2D human body, hand, facial, and foot keypoints on single images. OpenPose uses a multi-stage CNN network architecture, that predicts the 2D confident maps of body parts as well as the part affinity fields (PAFs)[4]. As shown in Fig.1(c), PAFs represent a set of flow fields that encodes unstructured pairwise relationships between body parts of people. 2D keypoints could be inferred from the confidence maps and PAFs. 3D keypoint detection could only be achieved by 3D triangulation from multiple single views. Given a single video, OpenPose is able to estimate plausible 2D pose of each frame independently. Other tools like AlphaPose could achieve even better results.

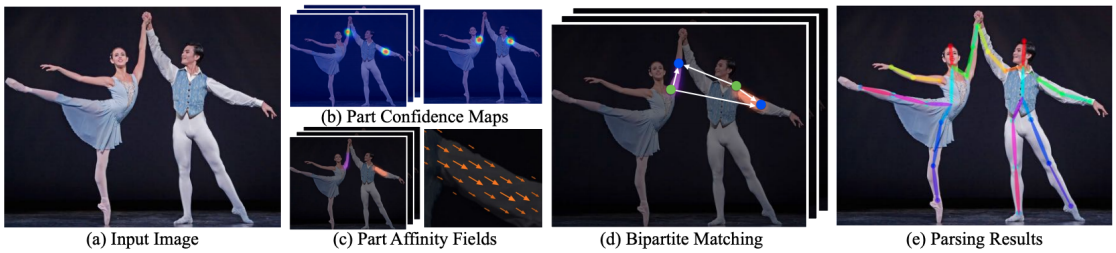

Fig. 1: (a) OpenPose takes the entire image as the input for a CNN to jointly predict (b) confidence maps for body part detection and (c) PAFs for part association. (d) The parsing step performs a set of bipartite matchings to associate body part candidates. (e) finally assemble them into full body poses for all people in the image

However, 3D pose estimation is still very challenging, due to lack of large-scale ground 3D annotation for in-the-wild image, and inherent ambiguities in single view 2D-to-3D mapping[10]. Temporal information of motions[7] and body shape would be helpful in this process.

VideoPose3D[18] is an efficient 3D human pose estimation in video making good use of motion sequence, the 2D keypoint trajectories are predicted from OpenPose. They designed a temporal dilated convolutional model with residual connections, which takes in a sequence of 2D poses and output 3D pose estimations, as shown in Fig.2. The training process is semi-supervised with both labeled 2D poses and unlabeled 2D poses. Given ground truth data, loss is computed from both the trajectories of human root joint and the Mean Per Joint Position Error(MPJPE) of all joints; As for unlabeled data, predicted trajectory and 3D poses are projected to 2D poses to compute 2D MPJPE as loss. They also add Bone length L2 loss that matches the mean bone lengths of subjects of the subjects of labeled batch as a soft constraint.

While VideoPose3D ignores the full-body shape, the estimated pose may not fit well when transfer to our humanoid characters. Human Mesh Recovery (HMR)[10] introduces an end-to-end adversarial learning of both human pose and shape. Human body is modeled as Skinned Multi-Person Linear (SMPL) model[16], including shape parameters $\beta \in R^{10}$, the first 10 coefficients of a PCA shape space, and pose parameters $\theta \in R^{3K}$ is modeled by relative 3D rotation of K = 23 joints in axis-angle representation. The key contribution of their work is to combine two unpaired large-scale datasets, one is 2D keypoint

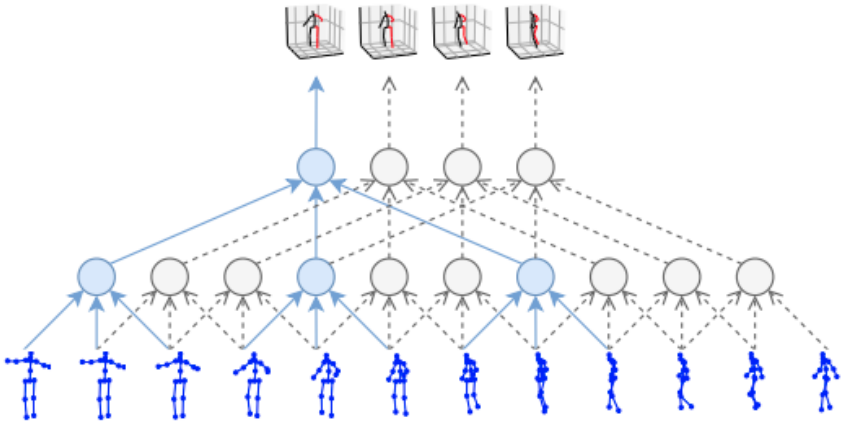

Fig. 2: VideoPose3D temporal convolutional model takes 2D keypoint sequences (bottom) as input and generates 3D pose estimates as output (top)to capture long-term information.

annotations of in-the-wild dataset, the other is 3D mesh of people with many poses. As shown in Fig. 3, their framework consists of a convolutional encoder that extracts features from the image, a regression module that infers the latent 3D representation of the human that minimizes the joint reprojection error, and a discriminator to tell if these parameters comes from a real human shape and pose. However, the major drawback is that they infer on each single frame independently, leading to discrete poses, which fail to produce smooth motion.

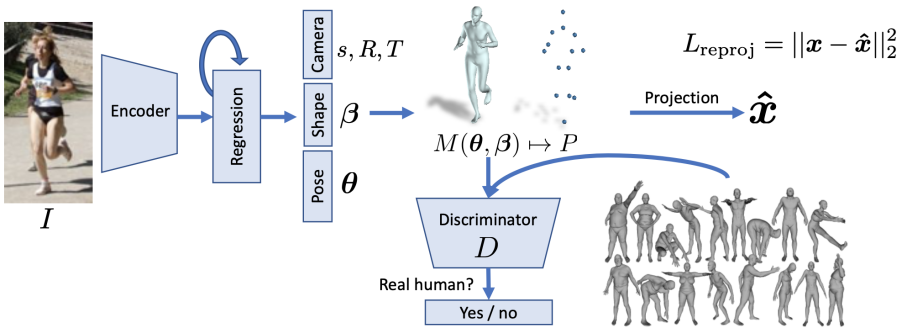

Fig. 3: Overview of HMR framework

A more recent model of 3D human pose estimation is VIBE[11], it get inspired by HMR[10], while trying to solve the problem of HMR. The goal is to produce realistic and accurate pose as well as shape, which refers to an existing large-scale motion capture dataset (AMASS)[17]. After extracting features from images, VIBE uses a temporal generation network to preserve some temporal information before regression.

2.2 Pose transfer

Once poses are obtained, the next important thing to consider is to transfer the pose to the character. Even the keypoints of the skeleton are the same, the body proportions are different. Animation retargeting is a feature that allows animations to be reused between characters. There are some useful addon tool in Blender for retargeting. [15] propose a human avatar representation scheme based on intrinsic coordinates, which are invariant to isometry and insensitive to human pose changes.

2.3 Reinforcement Learning

In the physical-based simulation environment, the character does not simply imitate the motion, it should actually learn to perform the skill. Reinforcement learning(RL) is an effective method for motion control task. RL is a kind of method that allows computational algorithms to learn from interactions by mapping states and rewards to actions. In a traditional reinforcement learning system, four main elements are identified: a policy, a reward signal, a value function and a model of the environment. The policy is defined by the agent to characterize the mapping from the states of environment to actions. The states are generally the function of all the history including the past rewards, past observation of the environment and past rewards from the environment. The reward signal shows how good an action is under a certain state of environment while the value function is the combination of all the future rewards. The policy also chooses the action that yields the highest value at each state. The model is the mimic of the environment defined by the agent to predict possible future situations.

Recently, more RL approaches are explored for character simulation. Value iteration methods are applied to develop kinematic controllers that can mimic motion clips[12]. The application of deep neural networks to building agents has improved the performance of simulating a series of challenging tasks[2]. Policy gradient methods have been shown to be more reliable in continuous control problem[21]. Since there exists a challenge in characterizing the reward function for natural movement with the absence of biomedical models and objectives to achieve natural simulated locomotion[13], additional objectives such as penalties to undesirable behaviors are used. Alternatively, recent RL methods based on motion capture imitation, such as GAIL[9], address such challenges by using data to induce an objective. Deepmimic[19] is a state of art RL method in character controlling that takes a character model with kinematic reference motion as input and enables the characters to imitate the reference motion by satisfying the task objectives.

3 Implementation

The whole pipeline of Reinforcement Learning of Motion from Videos is shown in Fig.4. Given a video, estimate the pose of each frame to reconstruct a motion,

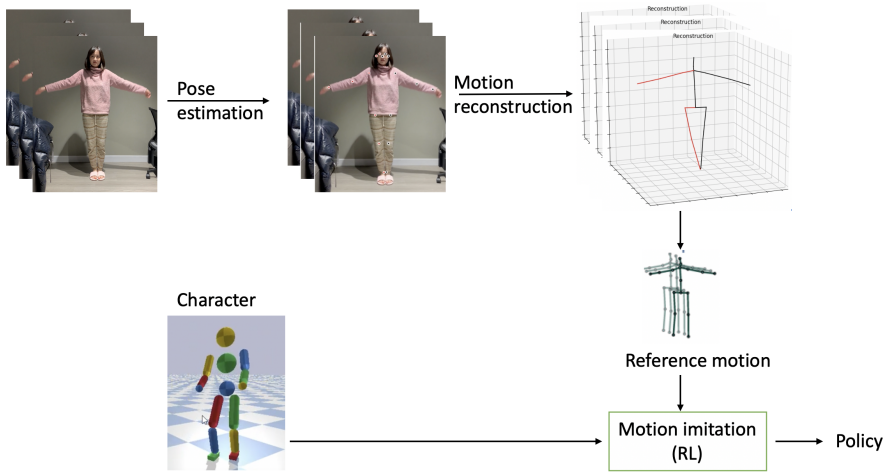

Fig. 4: Overview of Reinforcement Learning of Motion from Videos on videos

as the reference motion for character to imitate using Reinforcement Learning. In this project, I captured casually by my iPhone X. Most of the 3D pose estimations are based on the results of 2D pose estimation, 2D pose is necessary. With the popular tool OpenPose, we can easily obtain the 2D position of major keypoints of human. I implemented on Linux machine with GPU, CUDA, cuDNN installed. The deep learning framework for OpenPose is Pytorch and Caffe2. Results are shown in Fig.5. VideoPose3D is able to estimate the

Fig. 5: OpenPose output

With the keypoints positions, there are several ways to achieve 3D pose estimation. I implemented two recent approach, VideoPose3D and VIBE framework, and compare the results.

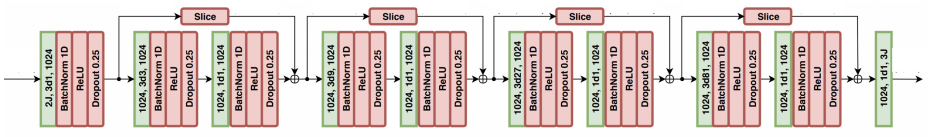

Fig. 6: VideoPose3D network architecture

Similar to OpenPose, VideoPose3D is implemented with Pythorch and Caffe2, the network archiecture is shown in Fig.6. The pose skeleton I used is COCO, which consists of 17 joints (Fig.7.). The video stream I used is extracted into 161 consecutive frames as a receptive field with J=17 joints, thus the input dimension for the network is (161, 34) .Convolutional layers are in green where 2J, 3d1, 1024 denotes 2·J input channels, kernels of size W=3 with dilation 1, and 1024 output channels. Then. tt is followed by B = 4 ResNet-style blocks, surrounded by a skip connection[8]. Each block first performs a 1D convolution with kernel size 1 and dilation3, then a convolution with kernel size 1. Convolutions are followed by batch normalization, rectified linear units, and dropout. Due to valid convolutions, we slice the residuals (left and right, symmetrically) to match the shape of subsequent tensors. Section 2.1 introduces the semi-supervised approach of VideoPose3D, the unlabeled data here is the 2D poses from the video and the labeled data is downloaded from Pretrained Human3.6m Coco Model. Some results are shown in Fig.8.

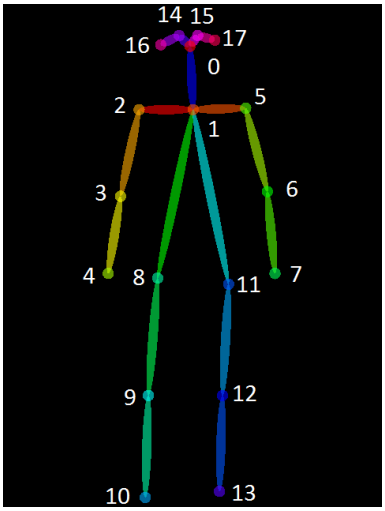

Fig. 7: Pose format of COCO

Although VideoPose3D adds Bone length L2 loss to compensate for the error of body shape, there may still be some problem without body shape estimation, leading to inaccurate 3D poses. VIBE is a more recent framework that developed from HMR. The generation part uses a temporal encoder, given consecutive frames $I_1, ..., I_160$, it first apply CNN to each frame to get features of 2048 dimensions, $f(I_1), ..., f(I_160)$. A 2-layer Gated Recurrent Unit (GRU) layer[6] yields a latent feature vector g_i for each frame, $g(f_1), ..., g(f_160)$. Then g_i are passed to regressor that outputs SMPL format results $\theta \in \mathcal{R}^{85}$. For simplicity, we use a gender-neutral shape model. The SMPL regressor has 2 fully-connected layers with 1024 neurons each. The proposed temporal encoder is composed of 2D(x), 3D(X), pose(θ) and shape(β) losses,

$$\mathcal{L}_{\mathcal{G}} = \mathcal{L}_{3D} + \mathcal{L}_{2D} + \mathcal{L}_{SMPL} + \mathcal{L}_{adv}, \tag{1}$$

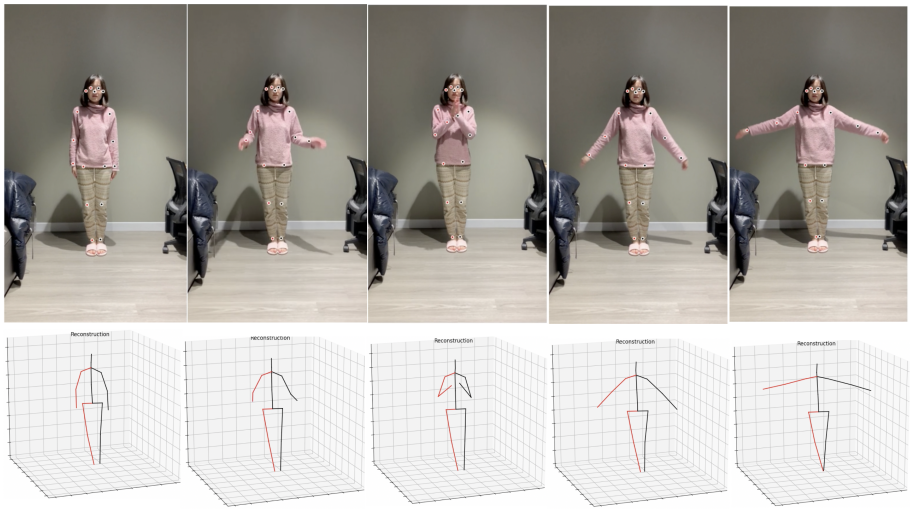

Fig. 8: VideoPose3D output

$$where \mathcal{L}_{3D} = \sum_{t=1}^{160} \left\| X_t - \widehat{X}_t \right\|_2, \mathcal{L}_{2D} = \sum_{t=1}^{160} \left\| x_t - \widehat{x}_t \right\|_2, \tag{2}$$

$$\mathcal{L}_{SMPL} = \left\| \beta_t - \widehat{\beta}_t \right\|_2 + \sum_{t=1}^{160} \left\| \theta_t - \widehat{\theta}_t \right\|_2 \tag{3}$$

Then the discriminator is applied to tell if the result of generator is a real world pose according to AMASS. $\mathcal{L}_{adv}$ is the adversarial loss.

$$\mathcal{L}_{adv} = \mathcal{L}_{\theta \sim P_G}[(\mathcal{D}_M(\widehat{\theta}) - 1)^2] \tag{4}$$

VIBE can be built in the same environment as VideoPose3D. Some results are shown in Fig.9. Now the question is which model generate better pose in our need? Our ideal pose sequence is accurate and smooth. Since VIBE outputs a body mesh that fits into the video, it guarantees more accuracy. To compare the smoothness of two outputs, we should transfer them into the same format. VideoPose3D outputs the 3D position of 17 joints, while VIBE outputs the rotation of 23 joints as well as the positions. Position is not a good representation of pose, as it will change with the proportion of body. Rotation is invariant to scale and body proportion, we represent the pose in the following format:

root position (3D), root rotation (4D), chest rotation (4D), neck rotation (4D), right hip rotation (4D), right knee rotation (1D), right ankle rotation (4D), right shoulder rotation (4D), right elbow rotation (1D), left hip rotation (4D), left knee rotation (1D), left ankle rotation (4D), left shoulder rotation (4D), left elbow rotation (1D).

Most joints are spherical joint in 4D quaternions. Then calculate the differential of each joints to show the angular velocity. Draw the histogram of angular velocity value of joints from the same video (Fig.10.). It seems that VideoPose3D

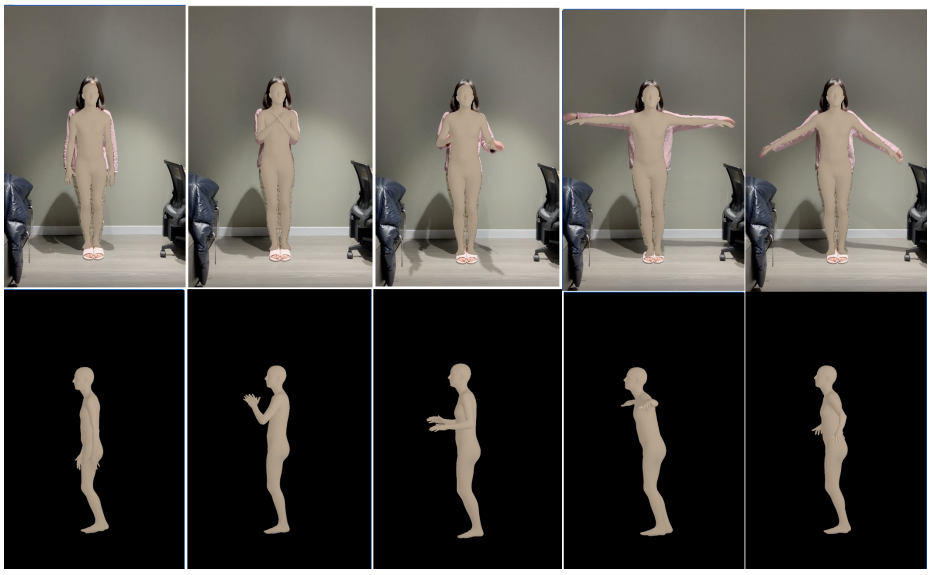

Fig. 9: VIBE output

values appear in lower ranges, showing that VideoPose3D motion is smoother in this example, but the differences are small. Thus We choose VIBE results for the following experiments.

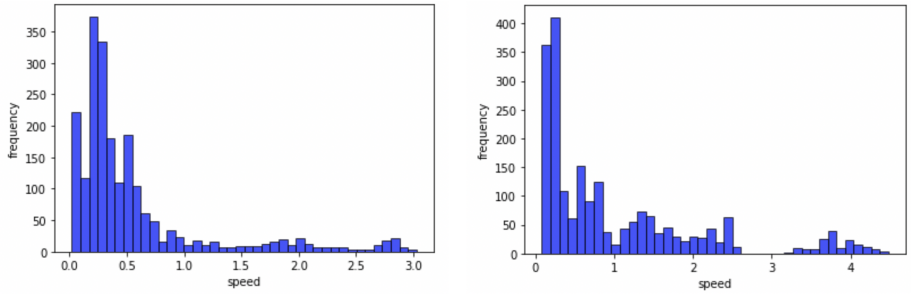

Fig. 10: Histogram of angular velocity value of joints

The simulation environment we use is pybullet, the python binding of Bullet Physics. Character configuration including shape and kinetics is declared before the simulation, as well as a stable PD controller[23]. The task is to imitate the reference motion. DeepMimic[19] defines the state s as relative positions of each link with respect to the root (designated to be the hip), their rotations expressed in quaternions, and their linear and angular velocities. To fill the gap between frames, linear interpolation is applied. The action a specifies target orientations for PD controllers at each joint. Policy is responsible for determining which actions should be applied at each timestep in order to reproduce the desired motion. Here the policy is defined as a feedforward network that receives as input

the current state s and outputs an action distribution $\Pi(a|s)$. It also proposed a variant of proximal policy optimization (PPO)[22] to train the policy.

In the training process, the initial state distribution $p(s_0)$ determines the states in which an agent begins each episode. The goal of the agent is to learn the optimal parameters θ that maximizes its expected return

$$\mathcal{J}(\theta) = E_{\tau \sim p_\theta(\tau)}[\sum_{t=0}^{T} \gamma^t r_t] \tag{5}$$

where $p_\theta(\tau) = p(s_0)\Pi_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi_t heta(a_t|s_t)$ is the distribution over all possible trajectories $\tau = (s_0.a_0, s_1, ..., a_{T-1}, s_T)$. $\sum_{t=0}^{T} gamma^t r_t$ is the total return of the trajectory. The reward is decomposed as

$$r_t^1 = w^p r_t^p + w^v r_t^v + w^e r_t^e + w^c r_t^c \tag{6}$$

$$w^p = 0.65, w^v = 0.1, w^e = 0.15, w^c = 0.1 \tag{7}$$

where The pose reward r_t^p encourages the character to match the joint orientations of the reference motion at each step,

$$r_t^p = exp[-2(\sum_j \left\| \widehat{q}_t^j - q_t^j \right\|^2)] \tag{8}$$

$\widehat{q}_t^j$ and q_t^j represent the orientations of the jth joint from the simulated character and reference motion respectively; The velocity reward r_t^v is computed from the difference of local joint velocities,

$$r_t^v = exp[-0.1(\sum_j \left\| \widehat{\dot{q}}_t^j - \dot{q}_t^j \right\|^2)] \tag{9}$$

The end-effector reward r_t^e encourages the character's hands and feet to match the positions from the reference motion, p_t^e denotes the 3D world position in meters of end-effector e (left foot, right foot, left hand, right hand),

$$r_t^e = exp[-40(\sum_e \|\widehat{p}_t^e - p_t^e\|^2)] \tag{10}$$

r_t^c penalizes deviations in the character's center-of-mass from that of the reference motion,

$$r_t^c = exp[-10(\|\widehat{p}_t - p_t\|^2)] \tag{11}$$

Early termination is triggered whenever the character falls. To test the effectiveness of DeepMimic model, I first implemented in mocap data it provides(the motion to learn is jumping). Since the training time for RL can be very long, I used Message Passing Interface (MPI) for parallel computing on Google cloud Virtual Machines with 8 worker process. After training for some time, I apply the policy to the character in pybullet environment. As shown in Fig.11

and Fig.12. The transparent character shows the reference motion without any physics settings, while the colorful, opaque one is the simulated character that move according to the policy we obtained through training. Fig.11 uses policy that has trained for 2 hours, Fig.12 has trained for one day. Fig.12 performs better than Fig.11, that it can complete the whole jump and comtinue to another jump, it can jump at most three time successively. However it is still not good enough, it still falls sometimes, given more time for training, it could do better.

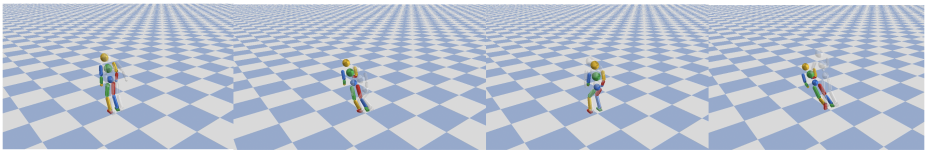

Fig. 11: DeepMimic on mocap data

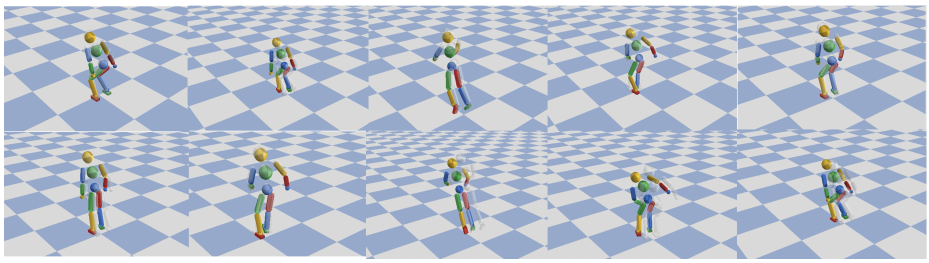

Fig. 12: DeepMimic on mocap data

The above results show that DeepMimic is a effective tool for motion control. I apply it to my previous pose results, trained for one day, the simulation is shown in Fig.13. Fig.14 shows that how reward increase in training process.However, it is not stable as the previous experiment, even though the motion is easier. The reason is obvious, the quality of input motion data cannot still cannot compare to mocap data. In the future, I hope to optimize the motion before RL training. as well as the RL framework, in order to get more satisfactory simulations.

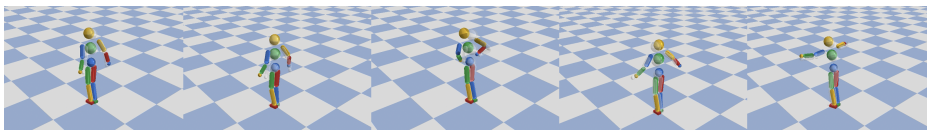

Fig. 13: DeepMimic on video

4 Conclusion

At the begining of this project, I was interested in DeepMimic paper, and I went to talk with Prof. Guy about the idea of transferring pose directly from daily videos. He encouraged to work on it nicely. However, the implementation didn't go smoothly. I got stuck several times, failing to plausible 3D poses, or failing to get DeepMimic working, what's worse, having difficulty to access computing resources in the lab due to the pandemic. I finally got the whole pipline to work. However, the current result is not the end. The simulation is unstable, and even

Fig. 14: DeepMimic rewards on each iteration

fails sometimes. I believe better pose estimation or better RL framework will lead to better simulation, and I sincerely hope that in the future I will be able to achieve that. Another limitation is that training takes too long, which makes the whole process slow. During this project, I learned a lot in Computer Vision and Reinforcement Learning. I'm extremely grateful to Prof. Guy, who encouraged me a lot, and gave me many great suggestions.

References

1. Bansal, A., Ma, S., Ramanan, D., Sheikh, Y.: Recycle-gan: Unsupervised video retargeting. In: ECCV (2018)
2. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym (06 2016)
3. Cao, Z., Hidalgo, G., Simon, T., Wei, S.E., Sheikh, Y.: Openpose: Realtime multi-person 2d pose estimation using part affinity fields (12 2018)
4. Cao, Z., Simon, T., Wei, S.E., Sheikh, Y.: Realtime multi-person 2d pose estimation using part affinity fields. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
5. Chan, C., Ginosar, S., Zhou, T., Efros, A.A.: Everybody dance now. In: IEEE International Conference on Computer Vision (ICCV) (2019)
6. Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: Conference on Empirical Methods in Natural Language Processing (EMNLP 2014) (2014)
7. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1243–1252. PMLR, International Convention Centre, Sydney, Australia (06–11 Aug 2017), `http://proceedings.mlr.press/v70/gehring17a.html`
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770–778 (2016)

9. Ho, J., Ermon, S.: Generative adversarial imitation learning. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29. pp. 4565–4573. Curran Associates, Inc. (2016), http://papers.nips.cc/paper/6391-generative-adversarial-imitation-learning.pdf
10. Kanazawa, A., Black, M.J., Jacobs, D.W., Malik, J.: End-to-end recovery of human shape and pose. In: Computer Vision and Pattern Regognition (CVPR) (2018)
11. Kocabas, M., Athanasiou, N., Black, M.: Vibe: Video inference for human body pose and shape estimation (12 2019)
12. Lee, Y., Wampler, K., Bernstein, G., Popović, J., Popović, Z.: Motion fields for interactive character locomotion. In: ACM SIGGRAPH Asia 2010 Papers. SIGGRAPH ASIA '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1866158.1866160, https://doi.org/10.1145/1866158.1866160
13. Lee, Y., Park, M.S., Kwon, T., Lee, J.: Locomotion control for many-muscle humanoids. ACM Trans. Graph. **33**(6) (Nov 2014). https://doi.org/10.1145/2661229.2661233, https://doi.org/10.1145/2661229.2661233
14. Levine, S., Popović, J.: Physically plausible simulation for character animation. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. p. 221–230. SCA '12, Eurographics Association, Goslar, DEU (2012)
15. Lifkooee, M.Z., Liu, C., Liang, Y., Zhu, Y., Li, X.: Real-time avatar pose transfer and motion generation using locally encoded laplacian offsets. Journal of Computer Science and Technology **34**, 256–271 (2019)
16. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: SMPL: A skinned multi-person linear model. ACM Trans. Graphics (Proc. SIGGRAPH Asia) **34**(6), 248:1–248:16 (Oct 2015)
17. Mahmood, N., Ghorbani, N., Troje, N.F., Pons-Moll, G., Black, M.J.: AMASS: Archive of motion capture as surface shapes. In: International Conference on Computer Vision. pp. 5442–5451 (Oct 2019)
18. Pavllo, D., Feichtenhofer, C., Grangier, D., Auli, M.: 3d human pose estimation in video with temporal convolutions and semi-supervised training. CoRR **abs/1811.11742** (2018), http://arxiv.org/abs/1811.11742
19. Peng, X.B., Abbeel, P., Levine, S., van de Panne, M.: Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. ACM Trans. Graph. **37**(4) (Jul 2018). https://doi.org/10.1145/3197517.3201311, https://doi.org/10.1145/3197517.3201311
20. Qian, X., Fu, Y., Xiang, T., Wang, W., Qiu, J., Wu, Y., Jiang, Y.G., Xue, X.: Pose-normalized image generation for person re-identification. In: The European Conference on Computer Vision (ECCV) (September 2018)
21. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 37, pp. 1889–1897. PMLR, Lille, France (07–09 Jul 2015), http://proceedings.mlr.press/v37/schulman15.html
22. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR **abs/1707.06347** (2017), http://arxiv.org/abs/1707.06347
23. Tan, J., Liu, K., Turk, G.: Stable proportional-derivative controllers. IEEE Computer Graphics and Applications **31**(4), 34–44 (2011)