Humanoid Robot Imitation of Human Motion from Instructional Videos: Paper Review and Test

Fatma Moalla
ENS Paris-Saclay & Ecole Centrale Paris
fatma.moalla@student.ecp.fr

Carla Yagoub
ENS Paris-Saclay & Ecole Centrale Paris
carla.yagoub@student.ecp.fr

Monday 20th January, 2020

Abstract

We will review and test "End-to-end recovery of human shape and pose" and "SFV: Reinforcement learning of physical skills from video" papers that propose a complementary approach to mimic human motions and poses from fixed camera videos by adding smoothness to it by using Deep Reinforcement Learning technique detailed in the second paper. In this final project within Object Recognition and Computer Vision class (MVA 19/20), we will first go through the latter models. Then we will apply the first model on different videos and evaluate the results. Therefore, we will use a pre-trained model for the reinforcement Learning and discuss its outcomes.

1. Introduction

Reconstructing human motion dynamics from real-life videos is a challenging and yet interesting problem. It has a major impact in many fields like self-driving cars (reproducing the motion of drivers in accidents), video games, clothing modelling and so on. Recent developments in human 2D/3D pose estimation from monocular images have been performed. As part of them, the end-to-end framework called Human Mesh Recovery (HMR) relies on some parametrization of human mesh [3]. However, the HMR framework produces many errors and returns unnatural physical motions, which proves the insufficiency of pose estimation to imitate human behaviors. That is why researchers from the University of Berkeley tackled this issue [6]. They coupled deep pose estimation with deep reinforcement learning to better capture and reproduce human motion.

The objective of the present paper is to handle and to reproduce some results found in [3] and [6]. In particular, we introduce the training of HMR models for the imitation of characters in personal videos by extracting their 3D motion. Then, we present some outputs of deep Reinforcement Learning (RL) pre-trained models to visualize the smoothness of 3D human motions. For both frameworks, we point out the limits of the models by specifically choosing challenging images/videos.

2. HMR model

2.1. First explanations

3D motion reconstruction has been addressed in many papers. The models they propose focus on predicting 3D joint locations by regression models based on their estimation of 2D joint detections. But these approaches are deemed insufficient by [3]. Indeed, the joints are sparse so they do not embody the whole human surface in 3D. Then, their locations do not capture all degrees of freedom of human mesh making impossible to infer the real pose. Finally, Kanazawa et al. argue that the inherent structure of the models estimating first 2D joint locations and then 3D model parameters from them is less adequate than a more straightforward manner. To tackle those limitations, they developped a model based on the parametrization of the 3D human mesh by shape and 3D joint angles. It uses the Skinned Multi-Person Linear (SMPL) model as encoding. This helps taking into account body proportions and deformations of body surface. The HMR model also directly computes 3D mesh parameters from image features without going through a multiple-stage process. This enables the context image to be considered in the model.

2.2. HMR stucture and loss function

The computation of the 3D pose relies on the minimization of the following loss function :
$L = \lambda(L_{reproj} + 1L_{3D}) + L_{adv}$
where L_{3D} is a loss term which should be included in case ground truth 3D data are available - in this paper, such annotations were not used in training. $L_{reproj} = \sum_i ||v_i(x_i\hat{x}_i)||_1$ is the joint reprojection error with $x_i \in R^{2K}$ ith ground truth 2D joints, $v_i \in 0, 1^K$ such that $v_i = 1$ if ith joint is visible, 0 otherwise for the K joints and $\hat{x}_i$ the output projection. This loss term is the part that makes the

3D body output match the 2D joint locations. As the reprojection loss is not sufficient to ensure that the SMPL parameters minimizing it correspond to a real body, they add a discriminator network which recognizes realistic from non-realistic output. The adversarial loss function L_{adv} represents that part of the network. λ is a control parameter that represents the trade-off between the different parts of the loss.

2.3. Openpose and pose keypoints

In order to better estimate the 3D mesh, the images need to be cropped to an input bounding box locating the target person. So, they are scaled to $224x224$ dimension and the diagonal of the bounding box is kept to 150px. This bounding box is computed by the model *openpose*. It returns the 2D locations of the human skeleton for each person. With these inputs, HMR model returns the 3D mesh in a video.

3. Reinforcement Learning Approach

3.1. Explanation of the DeepMimic Framework

In order to add smoothness to 3D human motions produced by the previous model, Deep Reinforcement Learning techniques are applied. In fact, after converting the kinematic motions of *openpose* to a robot configuration vector, DeepMimic module [5] takes these vectors and uses an agent/controller that interacts with an uncertain environment by following a learnt policy that aims to maximize the reward. More specifically, this reward characterizes how close the imitation is to the ground truth and how much the agent actions satisfies the targeted task objectives. After this policy learning step, the DeepMimic module [5] outputs a final Humanoïd robot model.

3.2. Policy training process

Following the previous approach, at each step t, a given agent in a state s_t takes the action a_t from a given parametric policy π_θ. As a result, this agent receives a reward r_t and gets a new state s_{t+1}. The ultimate goal of the policy training is to maximize, the expected return of an induced trajectory τ by π_θ, being the distribution over all trajectories: $J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[\sum_{t=0}^{T} \gamma^t r_t]$ with $\tau = (s_0, a_0, ..., a_{T-1}, s_T)$ and $\gamma \in [0,1]$ a discount factor. In order to optimize the objective function $J(\theta)$, we use the Proximal Policy Optimization (Appendix D).

As explained in the previous section, the reward r_t is characterized by two different components: a reward related to the imitation r_t^I that encourages the agent to follow the targeted motion and another reward r_t^G that pushes the controller/agent to obey and accomplish a specific task. Hence, the considered reward is the following: $r_t = w_I r_t^I + w_G r_t^G$ with w_I and w_G the respective weights related to imitation and task accomplishment.

4. Implementation on different datasets

4.1. HMR implementation

To better understand the structure of HMR and the use of 2D pose keypoints openpose outputs, simple tests were performed: on images then videos.

4.1.1 Installation steps

Before applying any model, the first task was to install modules requirements and packages.

- The HMR code [1] requires Python 2.7 with specific versions of *opendr:0.77* and *TensorFlow:1.3.0*

- The openpose code [4] requires a Python 3.6 environment with the specific package *Caffe* and building C++ codes with adequate *Cuda* and *CudNN* versions. After multiple attempts and successful installation of *Caffe* in *Ubuntu 18.04*, an error of incompatible computer architecture appeared. The problem was solved by going through *Google Colab*

4.1.2 Images of single person

The second step was to apply the HMR model on images datasets proposed in [1]. The *Barbell-0002* dataset was used. The openpose outputs were already available in a *pkl* format for the images. As the HMR code takes only *json* files as inputs, we have translated the keypoints in a readable dictionary. Once this was done, we launched the model with and without the openpose *json* files. We could then assess the importance of cropping the image. Only minor changes in sight are present with and without openpose (figures 6 and 7). We gathered the errors committed by HMR in the following table, as ground-truth data were available to give some quantitative comparisons.

Images	std Error	max distance	Error std data
1 - without	0.48	31.45	0.40
1 - with	0.53	30.20	0.49
132 - without	0.66	45.31	0.59
132 - with	0.58	32.35	0.53
263- without	0.47	32.25	0.39
263 - with	0.57	24.16	0.45

The errors were computed with *eval-video.py* in [1]. It solves the Procrustes problem between the ground-truth configuration of points and those provided by HMR (see figure C). To better analyze the results, a min-max standardization was used on the data. We can see that for some configurations, cropping seems worse considering the standard errors. But the maximum euclidean distance between pairs of points always decreases with cropping, hence its importance.

[1] https://github.com/akanazawa/hmr

4.1.3 Single person video

The third step was to apply the model on personal videos of less than 5 secs. Each person performed separately a simple task : jumping.

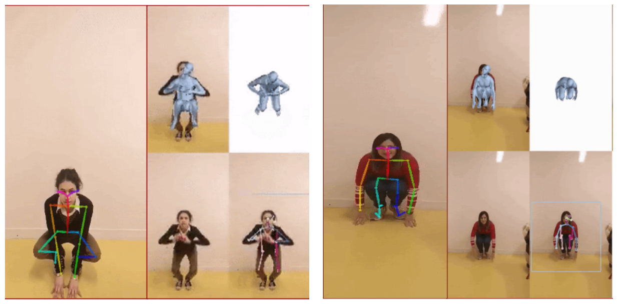


Figure 1. HMR output for personal videos

Even if openpose returned good estimations, we obtained less accurate predictions of the 3D mesh than for images. These differences can be imputed to the bad shooting and orientations of the camera.

4.1.4 Two people video

A final step relied on applying the model for more than one person video performing a dance battle [2], as this configuration is said to be challenging. We performed the pose estimation for 5 secs (2:55 to 3:00). We launched the model for 300 iterations (lasted 545s) and the convergence was assessed by the total loss function (see fig 5) as explained above.

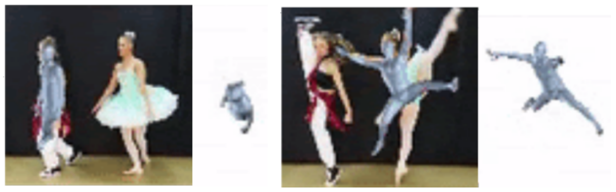


Figure 2. HMR output for dance battle at two moments of the video

As expected, HMR did not deal well with the two people and jumps from one to the other.

4.2. DeepMimic implementation

4.2.1 Installation steps

Again before applying any model, the first task was to install modules requirements and packages. After unsuccessful installations on personal OS (*MacOS Catalina 2018* and *Ubuntu 18.04*), DeepMimic [5] could finally be installed thanks to a *VirtualBox with Ubuntu 14.06* [7]. However, technical issues remained unsolved for the code $mpi\_run.py$, which prevented us from training any new configuration.

4.2.2 DeepMimic step

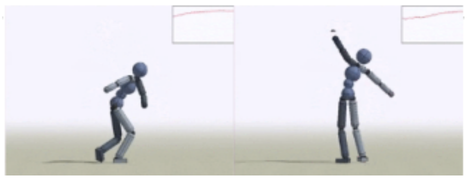


Figure 3. DeepMimic output for a dance configuration

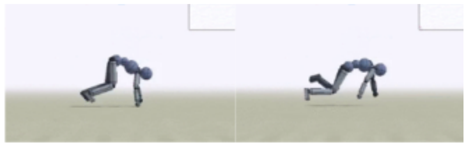


Figure 4. Run configuration applying different motions

We used a pre-trained model (DeepMimic github [5]) and we run the model for different configurations displayed in figure 3. For the dance configuration, Humanoïd robot accomplished the task perfectly using the pre-trained model. For the following step, with the same pre-trained models from [5] and using the pose *run*, we tried several motions in order to challenge the model. As displayed in figure 4, using motions like *jump* or *Get_up_face_down* leads to confusion in the *run* pose step-up.

5. Conclusion and perspectives

This project has taught us valuable aspects of research in computer vision both in theory and practice. We have also tested qualitatively and quantitatively the limits of the models for a better use.

As for the next steps, in addition to solving technical issues related to $mpi\_run.py$, we will train the model on the same videos used in HMR by looking for the best policy and we will add smoothness in motion reconstruction step by applying smoothness loss.

References

[1] Handtools dataset. Accessed: 2019-12-01.

[2] Ballet vs hip hop! video, 2018.

[3] A.Kanazawa. End-to-end recovery of human shape and pose. *CoRR*, abs/1712.06584, 2017.

[4] Zhe Cao. Openpose: realtime multi-person 2d pose estimation, 2018. Accessed: 2020-01-10.

[5] xbpeng. Deepmimic original github, 2019. Accessed: 2019-12-01.

[6] X.B.Peng, A.Kanazawa, J.Malik, P.Abbeel, and S.Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Trans. Graph.*, 37(6):178:1–178:14, Dec. 2018.

[7] zhaolongkzz. Deepmimic configuration github, 2019. Accessed: 2020-01-10.

A. Total Loss function for training HMR : dance battle

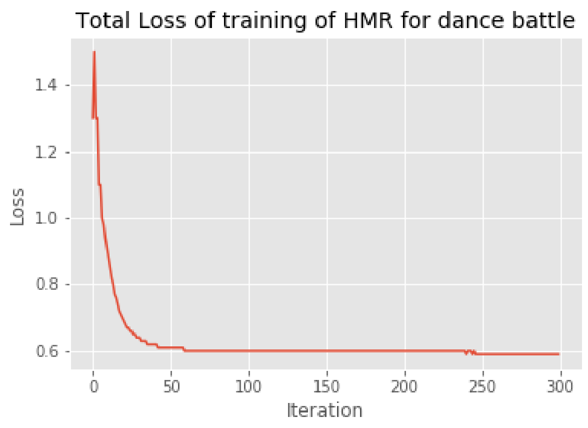


Figure 5. Total loss function for HMR training of the video Dance Battle

B. HMR output for Barbell image dataset

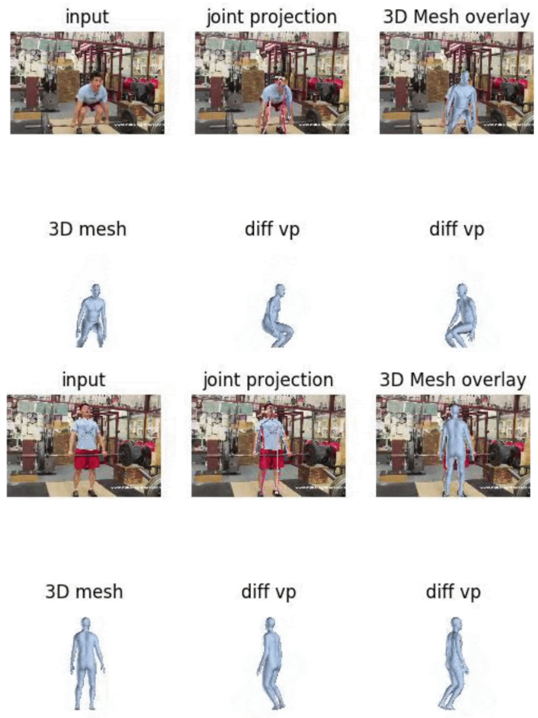


Figure 6. HMR output for Barbell images dataset without cropping

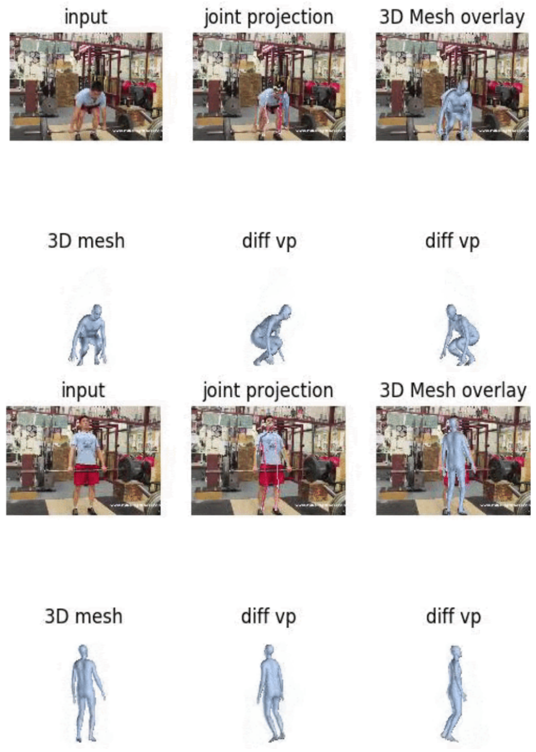
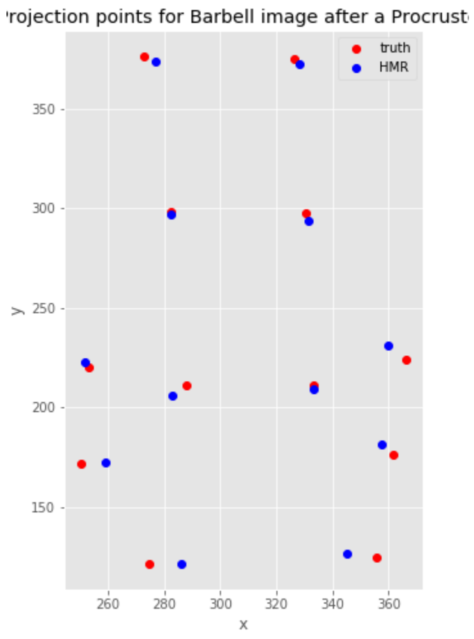


Figure 7. HMR output for Barbell images dataset with cropping

C. Evaluation of HMR results for Barbell image with Procrustes analysis



D. Proximal Policy Optimization algorithm

ALGORITHM 1: Proximal Policy Optimization

```
$\theta \leftarrow$ random weights
$\psi \leftarrow$ random weights
while not done do
    $s_0 \leftarrow$ sample initial state from reference motion
    Initialize character to state $s_0$
    for step = 1, …, $m$ do
        $s \leftarrow$ start state
        $a \sim \pi_\theta(a|s)$
        Apply $a$ and simulate forward one step
        $s' \leftarrow$ end state
        $r \leftarrow$ reward
        record $(s, a, r, s')$ into memory $D$
    end for

    $\theta_{old} \leftarrow \theta$
    for each update step do
        Sample minibatch of $n$ samples $\{(s_i, a_i, r_i, s'_i)\}$ from $D$

        Update value function:
        for each $(s_i, a_i, r_i, s'_i)$ do
            $y_i \leftarrow$ compute target values using TD($\lambda$)
        end for
        $\psi \leftarrow \psi + \alpha_v \left( \frac{1}{n} \sum_i \nabla_\psi V_\psi(s_i)(y_i - V(s_i)) \right)$

        Update policy:
        for each $(s_i, a_i, r_i, s'_i)$ do
            $\mathcal{A}_i \leftarrow$ compute advantage using $V_\psi$ and GAE
            $w_i(\theta) \leftarrow \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{old}}(a_i|s_i)}$
        end for
        $\theta \leftarrow$
        $\theta + \alpha_\pi \frac{1}{n} \sum_i \nabla_\theta \min(w_i(\theta)\mathcal{A}_i, \text{clip}(w_i(\theta), 1-\epsilon, 1+\epsilon)\mathcal{A}_i)$
    end for
end while
```