Topic E - Humanoid Robot Imitation of Human Motion from Instructional Videos

MBOUWE JOEL - Master MVA
joel-jores.mbouwe-nanmou@student.ecp.fr
April 4, 2020

1. Introduction

This paper summarises my work on the final project of the MVA course Object recognition and computer vision. I worked on the project about Humanoid Robot Imitation of Human Motion from Instructional Videos where the main goal was to enable a simulated character to perform skills presents on a reference video and this in a variant environnement.

Character animation which has applications in many fields (video games, robotics, 3D animation) can be significantly improved by the use of motion capture data but acquiring motion capture data is a challenging task because it requires important costs. Recents papers [4][8] have leveraged this issue by using directly monocular videos (videos from Youtube for instance). Those papers propose an end to end pipeline that processes a video and outputs a robust policy that can be used by a controller to imitate the skills in the video.

My main contribution is making an end to end connection between all the frameworks that are used to achieve the initial goal, testing the framework on the handtool dataset videos and analysing both qualitatively and quantitatively the results.

2. Overview of the Pipeline

The article [8] proposed this pipeline 1 that takes as input a video and a simulated character, estimates the 2D and 3D pose of the persons presents in the videos, computes a motion trajectory that is consistent with both 2D and 3D estimated. Then, a policy [7] is trained on the character in order to replicate the motion trajectory in a variant environment.

2.1. Pose estimation

2.1.1 2D pose estimation

Given a image, the objective is to identify joints, which are the coordinates of the key points that characterize a human (in this case 19 joints) of all the persons on the image, and

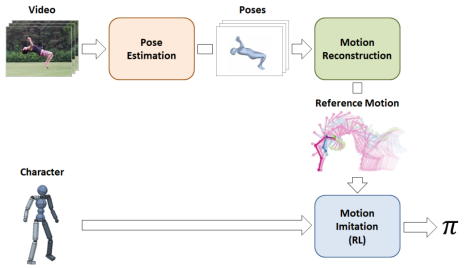

Figure 1: Pipeline

to find the links between those joints (for instance the hip is linked to the pelvis). For this task the library openpose [3] is used to perform both detection and 2d pose estimation. The process of estimating 2d pose in openpose consists in 3 steps : first extracting the features on the image by using the VGG first layers, second finding key points locations with some confidence score and finally pairing the key points. The figure 2 shows the ouptut of openpose on a couple of

Figure 2: 2D pose illustration

images. As it can been seen, the model works quite well even when some parts of the body are hidden.

2.1.2 3D pose estimation

The goal here is to reconstruct a full 3D mesh of a human body directly from an input image. For this task [4] proposes the framework in Figure 3 In this approach the model is trained in a weakly supervised way which is without ground truth 3D annotations but instead it uses 2d an-

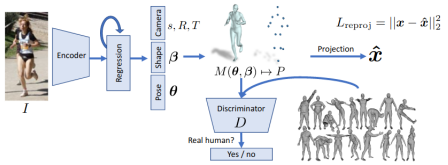

Figure 3: 3D pose

notations and the training consists in minimizing the joint reprojection L-2 error. Also, it uses and Adversarial discriminator in order to tell if the predicted pose and shape come from a real human shape and pose. From an input image, it outputs the Camera position, the Body shape and the Joint angles which by using the SMPL leads to the 3D mesh.

Besides, because we are trying to imitate acrobatics movements which are underrepresented in existing datasets, [8] proposes an augmentation dataset approach which consists in rotating some existing images and points out that it significantly improves the predicted poses of acrobatics movements. Below, Figure 4 is the ouptut of the hmr model applied to images.

Figure 4: 3D pose illustration. The estimated poses seem to fit perfectly the original frames

2.2. Motion Reconstruction

The objective is to reconstruct the original motion as a motion trajectory using both 2D and 3D poses. Because we are dealing with acrobatics videos which may include abrupt changes between consecutives poses; simply sequencing the 3D poses can produces awkward or unrealistic results. In order to address this issue, [8] proposes to define a smoothness criterion in order to enforce the temporal consistency between consecutives frames. Therefore the training consists in minimizing the loss $L_{rec} = w_{2D}l_{2D} + w_{3D}l_{3D} + w_{sm}l_{sm}$ where l_{2D} is the reprojection error between the predicted 2D joint locations, the 2D projections of the corresponding joints l_{3D} measures the 3D consistency and l_{sm} is the loss related to smoothness.
Note: it was very difficult to install openpose and all the librairies required by the repository motion_reconstruction [1] due to code errors, problems related to cuda, version of python etc.

2.3. Motion Imitation

The goal is to learn a policy with which a controller can imitate the behaviour of the original video and in a simulated environment. In this setting, the state space represents the position of the joints and the action space consists of the set of orientations of the controller at each joint. The action and value functions are modeled here by fully connected layers. The learning process is done by using Deep Reinforcement Learning[7] where the goal is to maximise the expected return : $E_{\tau \sim p_\theta}\left[\sum_t \gamma^t r_t\right]$ and r_t is defined by $r_t = w^p r_t^p + w^v r_t^v + w^e r_t^e + w^c r_t^c$ and is set so that the character matches the reference motion in joints orientation, joints velocity, hand and feet position and center of mass position.

Furthermore [8] proposes an approach in order to improve the speed and performance of the learning process. It proposes to define one more agent which goal will be to find the better state at which the controller might start at each episode.

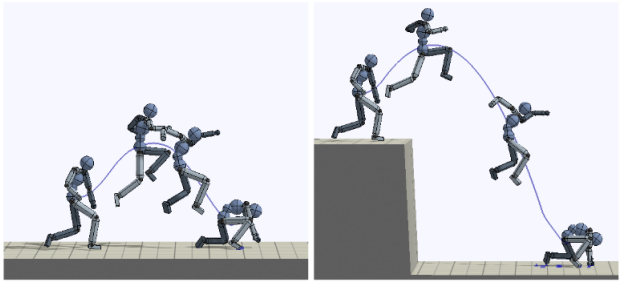

Fig. 10. **Left:** Original landing motion on flat terrain. **Right:** Policy trained to imitating landing motion while jumping down from a $2m$ ledge. Despite being provided only with a motion recorded on flat terrain, the policy is able to adapt the skill to jump down from a tall ledge.

Figure 5: Source : DeepMimic [7]

3. Application & Results

3.1. Experimental configuration

For this project, i worked on both google colab for the motion reconstruction part and my personal computer for motion imitation. Installing all the required packages was a challenging task. I spent more than one week figuring out how to install both openpose [3] and motion reconstruction

[1]. and a couple of days building deep_mimic [7]. The google colab notebook and the local script i used for the project can be found here [6].

For the analysis below, i used the handtools dataset [6] and because we have the ground truth 3D poses, i will do quantitative evaluations of the hmr[4] model with and without the smoothness approach. Furthermore i will also compare the 2D pose estimate between openpose and other pose library.

3.2. Analysis of the results

3.2.1 Pose Estimation

I applied both the hmr [4] model and a baseline 3d pose model [5] on the images of the handtool dataset. And the table 1 can be found the mean per joint position error (MPJPE) computed on some frames of the videos. For computing this error i adapted the script eval_video located in the handtool folder.

video name	MPJPE baseline	MPJPE HMR
barbell_0002	67.8	26.8
barbell_0003	NaN	28.0
barbell_0007	NaN	29.0
barbell_0008	NaN	34.9
barbell_0010	60.6	22.5
hammer_0003	54.5	20.2
hammer_0006	53.3	20.3
hammer_0007	54.4	20.9
hammer_0010	57.6	29.1
scythe_0001	61.3	29.0
scythe_0002	67.9	22.0
scythe_0003	58.7	21.1
scythe_0005	79.9	27.7
scythe_0006	60.2	24.3

Table 1: Mean per joint position error for a baseline model and hmr.

The hmr algorithm outperforms the baseline model. Indeed the baseline only uses the 2d pose to predict the 3D poses and no longer uses the original input, therefore it looses some informations. Meanwhile the hmr on the contrary, infer 3D mesh parameters directly from image features by using a regressor and refines the predicted 3D mesh using the 2D key points.

Also, it is clear that the MPJE of the model without the smoothing step will be slightly lower than the one with the smoothing.

3.2.2 Motion Reconstruction

Evaluating quantitatively the motion reconstruction part was impossible in this project, instead i did a qualitative evaluation. And Figure 6 shows the reconstruction video obtained. (All the videos can be found here [6])

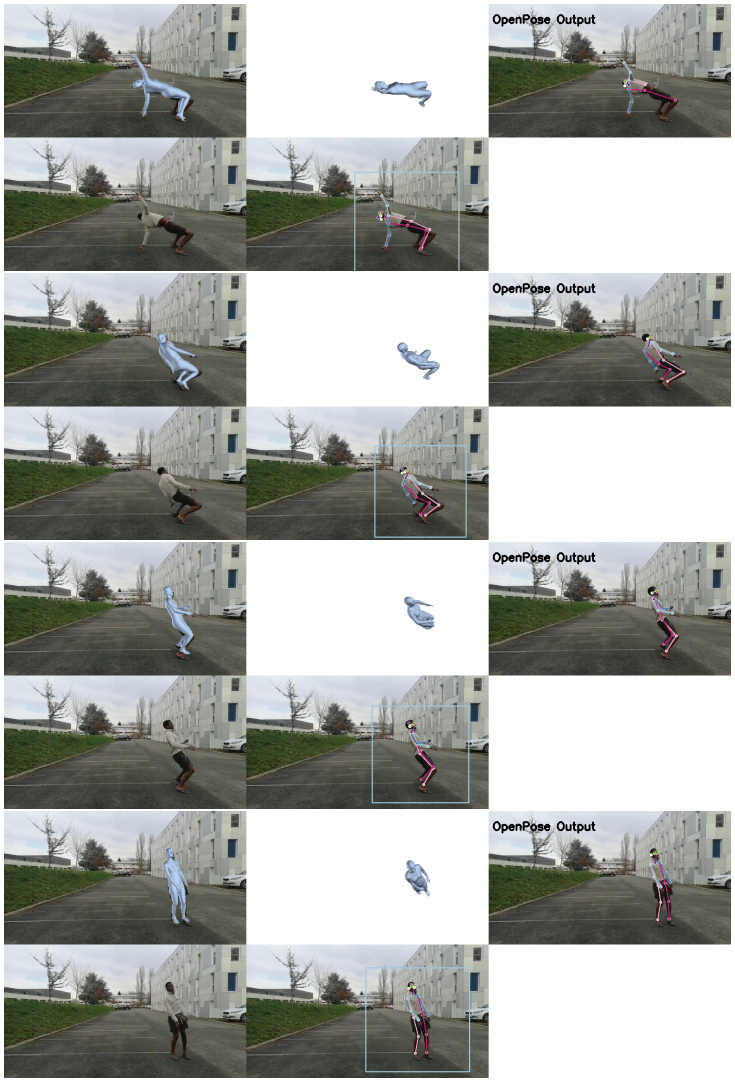

Figure 6: From top to down, visualizations of the HMR-smoothed results for some frames. It shows: Top Left: Recovered mesh overlayed, Top Center: From a different view, Top Right: Results of OpenPose, Bottom Left: Input video, Bottom Center: 2D joint reprojection of the recovered 3D mesh

As it can be seen on these frames, the results of openpose tends to be different from the 2D joint reprojection of the recovered 3D mesh and this difference enables the reconstruction step to avoid non physic movements and inconsistency between consecutives frames.

3.2.3 Motion Imitation

I manage to build the deep_mimic [7] package but i wasn't able to train my own policy. The main challenge after building the package was the mapping of the output of the HMR file as input motion for the controller. I tried an online library called bvhtodeepmimic but because the joints positions in hmr were computed by taking the root as origin, the lower body of the controller representing the motion trajectory build with bvhtodeepmimic was in the ground.

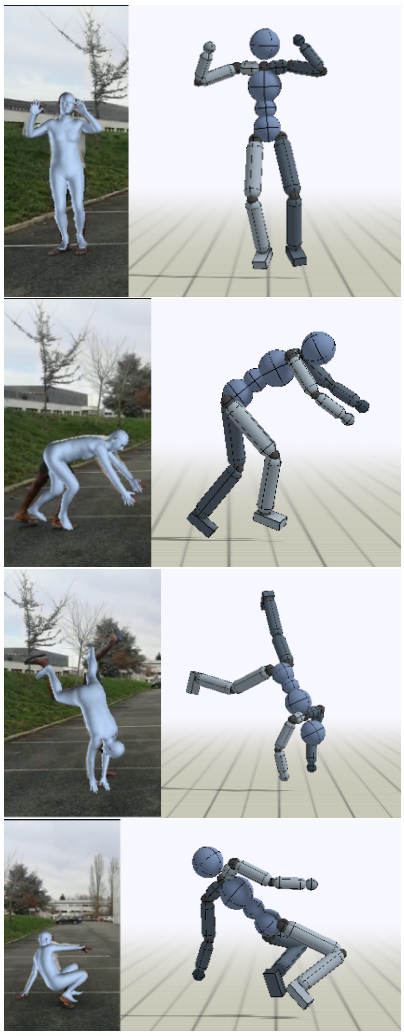

Figure 7: motion reference run with the humanoid character. Left is the 3D mesh and. Right is the controller imitating the motion

Fortunately, i ran into a code [2], written by a former MVA student and i was then able to convert effectively the output of hmr to deep_mimic even if for some of the motions, the feet of the controller were floating. I was therefore able to visualize 7 a reference motion however i came across an error when trying to learn my own custom policy : ¡ Invalid path data value detected ¿ and i couldn't solve the problem. On Figure 7, can be found a reference motion video applied to a humanoid controller.

4. Conclusion

The project was to some extent very interessant in the way it deals with both computer vision, reinforcement learning. Unfortunately i couldn't achieve the integrality of what i wanted because of the long time i spent getting the code to work. Indeed the documentations we had could have be better. When i will have some free time, i will make sure to solve the problem raised by deep_mimic when trying to train a new policy.

References

[1] akanazawa. URL: `https://github.com/akanazawa/motion_reconstruction`.

[2] Gilles Bareilles. URL: `https://github.com/GillesBareilles/MVA_ORCV_Presentation`.

[3] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.

[4] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Computer Vision and Pattern Regognition (CVPR)*, 2018.

[5] Julieta Martinez, Rayat Hossain, Javier Romero, and James J. Little. A simple yet effective baseline for 3d human pose estimation. In *ICCV*, 2017. URL: `https://github.com/ArashHosseini/3d-pose-baseline`.

[6] Joel Mbouwe. URL: "`https://github.com/Ryosaeba8/final_project_recvis19`".

[7] Levine Peng, Abbeel and Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. 2018.

[8] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. 2018.