TEMOS: Generating diverse human motions from textual descriptions Supplementary Material

Mathis Petrovich[1,2], Michael J. Black[2], and Gül Varol[1]

[1] LIGM, École des Ponts, Univ Gustave Eiffel, CNRS, France
[2] Max Planck Institute for Intelligent Systems, Tübingen, Germany
{mathis.petrovich,gul.varol}@enpc.fr, black@tue.mpg.de
https://mathis.petrovich.fr/temos/

This appendix provides additional experiments (Section A), description of the motion representations (Section B), details on evaluation metrics (Section C), implementation details (Section D), and dataset statistics (Section E).

Video. Additionally, we provide a supplemental video, available on our website [10], which we encourage the reader to watch since motion is critical in our results, and this is hard to convey in a static document. In the video, we illustrate: (i) comparison with previous work, (ii) training with skeleton versus SMPL data, (iii) diversity of our model, (iv) generation of variable size sequences, (v) interpolation between two texts in our latent space, and (vi) failure cases.

Code. We also share our code base on the project page [10], which reproduces our training and evaluation metrics. Explanations on how to configure the data and launch the training can be found in the file `README.md`.

A Additional experiments

We conduct several experiments to explore the sensitivity of our model to certain hyperparameters. Our final set of hyperparameters was chosen using the validation set (starting from a set of hyperparameters similar to ACTOR [11]). Note that these hyperparameters did not always appear optimal on the test set. The following ablations provide a sense of robustness to different parameters. In particular, we show the effects of the batch size (Section A.1), $\{\lambda_{\text{KL}}, \lambda_{\text{E}}\}$ loss weighting parameters (Section A.2) and the architecture parameters of Transformers (Section A.3). All other parameters, such as the learning rate (10^{-4}), the optimizer (AdamW) and the number of epochs (1000) are fixed. For all these experiments we use the skeleton-based MMM representation.

We also experiment with various pretrained language models (Section A.4).

Note that the evaluation is based on a single random sample. All results can be improved by taking the best sample closest to the ground truth out of multiple generations (as next explained in Section A.5). We also experiment with taking the sample farthest from the ground truth.

Finally, in Section A.6, we report quantitative results for our model trained with SMPL rotations.

A.1 Batch size

In Table A.1, we present results with batch sizes of 8, 16, 24 and 32. To handle variable-length training, we use padding and masking in the encoders and the decoder. To maintain reasonable memory consumption, we discard training samples that have more than 500 frames (after sub-sampling to 12.5 Hz): this corresponds to about 2.3% of the training data. We show that we obtain the best results by setting the batch size to 8 or 32. We set it to 32 in all other experiments as it takes less time to train.

A.2 Weight of the KL losses and the embedding loss

In Table A.2, we report results by varying both λ_{KL} and λ_{E} parameters (described in Section 3.3 from 10^{-3} to 10^{-8}. We show that overall the results are similar when λ_{E} is fixed. A too high value of 10^{-3} deteriorates the performance. We fix both of them to 10^{-5}.

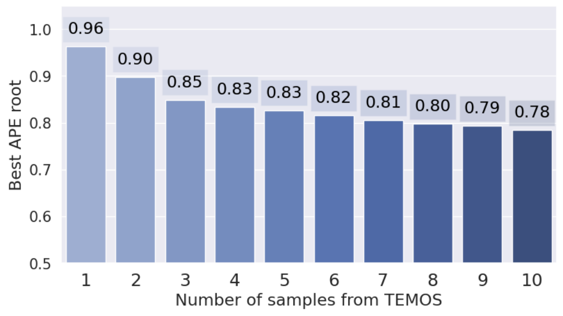

Fig. A.1: **Best APE root when sampling multiple generations:** Given a textual description, we generate multiple different motions, and select the motion that matches best to the ground truth sequence. We show that by sampling more generated sequences per text, we can reduce the APE root metric error.

Table A.1: **Batch size:** We see that the performance is the best for either a small batch size (=8) or a bigger batch size (=32). We were unable to use a higher batch size due to the GPU memory limit.

Batch size	Average Positional Error ↓				Average Variance Error ↓			
	root joint	global traj.	mean local	mean global	root joint	global traj.	mean local	mean global
bs = 8	**0.950**	**0.941**	0.105	**0.965**	0.449	0.448	0.005	0.451
bs = 16	1.115	1.106	0.105	1.128	0.513	0.512	0.005	0.515
bs = 24	1.260	1.250	0.106	1.273	0.542	0.542	0.005	0.545
bs = 32	0.963	0.955	**0.104**	0.976	**0.445**	**0.445**	0.005	**0.448**

Table A.2: **Weight of the KL losses ($\lambda_{\mathbf{KL}}$) and the embedding loss ($\lambda_{\mathbf{E}}$):** The results are influenced more by changes in λ_{E} than in λ_{KL}, but otherwise if the values are not too low, the performances are similar. Note that the control row $\lambda_{\mathrm{KL}} = 10^{-5}, \lambda_{\mathrm{E}} = 10^{-5}$ is repeated in each block.

Losses weight	Average Positional Error ↓				Average Variance Error ↓			
	root joint	global traj.	mean local	mean global	root joint	global traj.	mean local	mean global
$\lambda_{\mathrm{KL}} = \lambda_{\mathrm{E}} = 10^{-3}$	1.219	1.210	0.111	1.230	0.555	0.554	0.006	0.556
$\lambda_{\mathrm{KL}} = \lambda_{\mathrm{E}} = 10^{-4}$	1.110	1.101	0.106	1.122	0.476	0.475	0.005	0.479
$\lambda_{\mathrm{KL}} = \lambda_{\mathrm{E}} = 10^{-5}$	**0.963**	**0.955**	**0.104**	**0.976**	**0.445**	**0.445**	0.005	**0.448**
$\lambda_{\mathrm{KL}} = \lambda_{\mathrm{E}} = 10^{-6}$	1.242	1.233	0.105	1.254	0.586	0.585	0.005	0.589
$\lambda_{\mathrm{KL}} = \lambda_{\mathrm{E}} = 10^{-7}$	1.034	1.025	0.108	1.049	0.488	0.487	0.005	0.491
$\lambda_{\mathrm{KL}} = \lambda_{\mathrm{E}} = 10^{-8}$	1.085	1.075	0.107	1.099	0.490	0.489	0.005	0.493
$\lambda_{\mathrm{KL}} = 10^{-5}, \lambda_{\mathrm{E}} = 10^{-3}$	1.293	1.284	0.107	1.305	0.631	0.631	0.005	0.635
$\lambda_{\mathrm{KL}} = 10^{-5}, \lambda_{\mathrm{E}} = 10^{-4}$	1.039	1.029	0.104	1.052	0.449	0.448	0.005	0.452
$\lambda_{\mathrm{KL}} = 10^{-5}, \lambda_{\mathrm{E}} = 10^{-5}$	**0.963**	**0.955**	**0.104**	**0.976**	**0.445**	**0.445**	0.005	**0.448**
$\lambda_{\mathrm{KL}} = 10^{-5}, \lambda_{\mathrm{E}} = 10^{-6}$	1.082	1.072	0.107	1.095	0.456	0.455	0.005	0.459
$\lambda_{\mathrm{KL}} = 10^{-5}, \lambda_{\mathrm{E}} = 10^{-7}$	1.018	1.008	0.106	1.031	0.464	0.463	0.005	0.467
$\lambda_{\mathrm{KL}} = 10^{-5}, \lambda_{\mathrm{E}} = 10^{-8}$	1.076	1.067	0.105	1.089	0.477	0.476	0.005	0.480
$\lambda_{\mathrm{KL}} = 10^{-3}, \lambda_{\mathrm{E}} = 10^{-5}$	1.145	1.135	0.111	1.157	0.507	0.506	0.006	0.509
$\lambda_{\mathrm{KL}} = 10^{-4}, \lambda_{\mathrm{E}} = 10^{-5}$	1.070	1.061	0.106	1.083	0.471	0.470	0.005	0.474
$\lambda_{\mathrm{KL}} = 10^{-5}, \lambda_{\mathrm{E}} = 10^{-5}$	**0.963**	**0.955**	**0.104**	**0.976**	**0.445**	**0.445**	0.005	**0.448**
$\lambda_{\mathrm{KL}} = 10^{-6}, \lambda_{\mathrm{E}} = 10^{-5}$	0.971	0.962	0.105	0.986	0.455	0.454	0.005	0.457
$\lambda_{\mathrm{KL}} = 10^{-7}, \lambda_{\mathrm{E}} = 10^{-5}$	1.140	1.132	0.106	1.154	0.513	0.512	0.005	0.517
$\lambda_{\mathrm{KL}} = 10^{-8}, \lambda_{\mathrm{E}} = 10^{-5}$	1.025	1.015	0.107	1.039	0.461	0.460	0.005	0.464

Table A.3: **Number of layers and heads in all Transformers:** While our results are slightly better for larger models, we observe that the performance is not very sensitive to changes in the number of layers and heads in Transformers.

Transformers parameters	Average Positional Error ↓				Average Variance Error ↓			
	root joint	global traj.	mean local	mean global	root joint	global traj.	mean local	mean global
nheads = nlayers = 2	1.194	1.185	0.107	1.205	0.546	0.545	0.006	0.547
nheads = nlayers = 4	1.189	1.181	0.104	1.201	0.500	0.499	0.005	0.502
nheads = nlayers = 6	**0.963**	**0.955**	0.104	**0.976**	**0.445**	**0.445**	0.005	**0.448**

A.3 Number of Transformer layers

Here, we present two different experiments. The first experiment is to change globally the number of layers and the number of heads of all our Transformers. In Table A.3, we see that the results are optimal when they are both fixed at 6, which is used in all other experiments.

Next, in Table A.4, we experiment with a lighter model on top of the DistilBERT text encoder, by adding fewer layers and heads than 6. We see that 1 or 2 layers are not sufficient, but beginning with 4, the results are satisfactory. We use 6 layers in our model.

A.4 Pretrained language models

We experiment with replacing DistilBERT [14] with a larger pretrained language model. We compare with the original BERT [2] model as well as the more recent RoBERTa [7] model. The results are similar and suggest that DistilBERT is sufficient for this task, while having fewer parameters.

Table A.4: **Number of layers and heads in the Transformer of the text encoder only:** We fix the Transformer layers and heads of the motion encoder and motion decoder to 6 (as in the other experiments), but we only change the number of layers and heads of the *text encoder* (the one on top of DistilBert). The results suggest that training a light model on top of the language model still gives descent results, but adding more layers helps.

Transformers parameters	Average Positional Error ↓				Average Variance Error ↓			
	root joint	global traj.	mean local	mean global	root joint	global traj.	mean local	mean global
nheads = nlayers = 1	1.163	1.151	0.110	1.175	0.529	0.528	0.006	0.531
nheads = nlayers = 2	1.170	1.161	0.107	1.182	0.452	0.451	0.005	0.454
nheads = nlayers = 3	1.094	1.085	0.105	1.106	0.474	0.473	0.005	0.476
nheads = nlayers = 4	**0.916**	**0.908**	**0.104**	**0.930**	**0.440**	**0.440**	0.005	**0.444**
nheads = nlayers = 6	0.963	0.955	**0.104**	0.976	0.445	0.445	0.005	0.448

Table A.5: **Language model:** We experiment with language models larger than DistilBERT and do not observe significant changes in the performance.

Language model	Average Positional Error ↓				Average Variance Error ↓			
	root joint	global traj.	mean local	mean global	root joint	global traj.	mean local	mean global
DistilBERT [14]	**0.963**	**0.955**	**0.104**	**0.976**	0.445	0.445	0.005	0.448
BERT [2]	0.986	0.977	0.105	1.000	**0.441**	**0.441**	0.005	**0.444**
RoBERTa [7]	1.066	1.056	0.107	1.079	0.492	0.491	0.005	0.494

A.5 Sampling multiple motions

Given a text input, instead of generating a single motion as in previous methods [1, 4, 6], we can generate multiple motions. As demonstrated in Table 2 of the main paper, we can improve the evaluation metrics by picking the best out of a set of motion generations, that is closest to the ground truth. In Figure A.1, we plot the reduction in APE root error as we increase the number of generations per text from 1 to 10, and observe a monotonic decrease as expected.

Furthermore, we measure the *worst case scenario*, where we generate 10 motions per text and record the error between the ground truth motion and the most different generated motion out of the 10. We obtain an APE of 1.24 (instead of 0.78 in the best case scenario, and 0.96 in the random scenario). Note that calling this worst case may not be accurate since the single ground truth motion does not represent the only possible motion, i.e., our generations may correspond well to the text without being close to the ground truth joints.

A.6 Quantitative results with the SMPL model

To evaluate quantitatively our SMPL-based model, and obtaining results comparable with the MMM framework, we extract the most similar skeleton subset from the SMPL-H joints (provided by AMASS). The correspondence can be found in Table A.6.

Both in SMPL-H and MMM, the bodies are canonicalized with a standard body shape (robot-style for MMM, and average neutral body for SMPL-H). We further rescale the SMPL joints with a factor of 0.64, to match them with MMM joints. We evaluate the model with and without this rescaling. The performance

Table A.6: **Correspondence** between the SMPL-H joints and the MMM framework joints.

Type	Joints										
MMM SMPL-H	root pelvis	BP spine1	BT spine3	BLN neck	BUN head	LS left_shoulder	LE left_elbow	LW left_wrist	RS right_shoulder	RE right_elbow	RW right_wrist
MMM SMPL-H	LH left_hip	LK left_knee	LA left_ankle	LMrot left_heel	LF left_foot	RH right_hip	RK right_knee	RA right_ankle	RMrot right_heel	RF right_foot	

Table A.7: **Our results with SMPL model:** We evaluate our model trained on the SMPL data against the ground truth of the test set of $\mathrm{KIT}_{\mathrm{SMPL}}$ (joints extracted from the AMASS dataset). The first row shows results after a rescaling (to get skeletons closer to MMM processed joints). The second row shows results with the same set of joints but without the final rescaling. Both results are in meters.

		Average Positional Error ↓				Average Variance Error ↓			
Dataset	rescaled	root joint	glob. traj.	mean loc.	mean glob.	root joint	glob. traj.	mean loc.	mean glob.
$\mathrm{KIT}_{\mathrm{SMPL}}$	✓	0.698	0.689	0.091	0.712	0.157	0.157	0.004	0.161
$\mathrm{KIT}_{\mathrm{SMPL}}$	✗	1.097	1.077	0.169	1.118	0.384	0.383	0.009	0.393

metrics on KIT_{SMPL} test set can be found in Table A.7. Performance is comparable to that of MMM-based training when evaluating with rescaled joints. We expect future work to compare against the non-rescaled version when employing the SMPL model to interpret the metrics with respect to real human sizes.

B Motion representation

Here, we describe in detail the two motion representations employed in this work: skeleton-based (Section B.1) and SMPL-based (Section B.2). We standardize both of them by subtracting the mean and dividing with the standard deviation across the training data.

B.1 Skeleton-based representation

We employ the representation introduced by Holden et. al. [5]. As input, we use the 21 joints from the Master Motor Map (MMM) framework [15]. For each frame, we encode:

i) the **3D human joints** in a coordinate system local to the body (the definition of the 3 axes and the origin is explained in Figure A.2) without the root joint (20x3=60-dimensional features),
ii) the **angles** between the local X-axis and the global X-axis (by storing them as differences between two frames) (1-dimensional feature),
iii) the **translation**, as the velocity of the root joint in the body's local coordinate system for X and Y axes, and the position of the root joint for the Z axis (3-dimensional features).

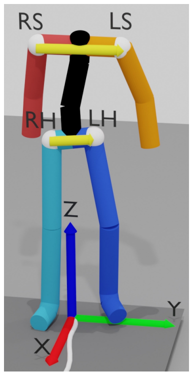

Fig. A.2: **Body's local coordinate system:** The origin is defined as the projection of the root joint into the ground. The X-axis direction is computed by taking the cross product between the average of the two yellow vectors: (left hip (LH) - right hip (RH)) and (left shoulder (LS) - right shoulder (RS)). The Z-axis (gravity axis) remains the same, and the Y-axis is the cross product of Z and X.

Then, we concatenate this into a feature vector in $\mathbb{R}^{64}$. For a motion sequence of duration F, the data sample will be in $\mathbb{R}^{F\times 64}$.

Note that, as we store the difference of angles for ii), when we integrate the angles, we assume that the first angle is 0 (which means that the body's local coordinate system is aligned with the global one). This is a way of canonicalizing the data so that each sequence starts with the body oriented in the same way.

B.2 SMPL-based representation

To construct the feature vector for SMPL data, we store:

i) the **SMPL local rotations** [8] (parent-relative joint rotation according to the kinematic tree) in the 6D continuous [18] representation (we load the AMASS data processed with SMPL-H [13], but we remove all rotations on the hands, resulting in 21 rotations) (21x6=126-dimensional features),
ii) the **global rotation** in the 6D continuous [18] representation (6-dimensional features),
iii) the **translation**, as the velocity of the root joint in the global coordinate system for the X and Y axes, and the position of the root joint for the Z axis (3-dimensional features).

Then, we concatenate this into a feature vector in $\mathbb{R}^{135}$. For a motion sequence of duration F, the data sample will be in $\mathbb{R}^{F\times 135}$.

To help the network learn the global rotation, we canonicalize the global orientation (by rotating all bodies and trajectories) so that the first frame is oriented in the same direction for all sequences.

C Evaluation details

In this section, we give details on the evaluation metrics and explain our implementation. We further provide details on our human study.

Metrics definitions. As explained in Section 4.1 of the main paper, the metrics are not computed with the original 3D joint coordinates. All motions are first canonicalized (so that the forward direction of the body faces the same direction for all methods and the ground truth). Except for the root joint, all the other joints are expressed in the body's local coordinate system (see Figure A.2). Then, the APE and AVE metrics are computed separately on these transformed coordinates.

As in JL2P [1] and Ghosh et al. [4], we define the evaluation metrics as follows, and compute them on the test set. The Average Position Error (APE) for a joint j is the average of the L2 distances between the generated and ground-truth joint positions over the time frames (F) and test samples (N):

$$\mathrm{APE}[j] = \frac{1}{NF} \sum_{n \in N} \sum_{f \in F} \left\| H_f[j] - \hat{H}_f[j] \right\|_2 . \tag{1}$$

We omit denoting the iterator n (over the test set) from the pose H for simplicity. The Average Variance Error (AVE), introduced in Ghosh et al. [4] captures the difference of variations. It is defined as the average of the L2 distances between the generated and ground truth variances for the joint j:

$$\mathrm{AVE}[j] = \frac{1}{N} \sum_{n \in N} \| \sigma[j] - \hat{\sigma}[j] \|_2 \tag{2}$$

where,

$$\sigma[j] = \frac{1}{F-1} \sum_{f \in F} \left(H_f[j] - \widetilde{H}_f[j] \right)^2 \in \mathbb{R}^3 \tag{3}$$

denotes the variance of the joint j. $\widetilde{H}[j]$ is the mean of the joint throughout the motion.

We report in the tables:

i) the *root joint* errors by taking the 3 coordinates of the root joint,
ii) the *global trajectory* errors by taking only the X and Y coordinates of the root joint (it is the white trajectory on the ground in the visualizations),
iii) the *mean local* errors by averaging the joint errors in the body's local coordinate system,
iv) the *mean global* errors by averaging the joint errors in the global coordinate system.

Metrics implementation. We were unable to reuse the evaluation code of Ghosh et al [4] for two reasons: (i) the code did not reproduce the results in [1, 4], (ii) there is a bug in the evaluation script `src/eval_APE.py` line 249, the slicing for the trajectory loss is wrong (notably `y[:, 0]` instead of `y[:, :, 0]`; this issue was confirmed by the authors).

Furthermore, the function `fke2rifke` might also have slicing bugs (the implementation is the same in the codes of `JL2P` and `Ghosh et al.`) and is used indirectly for evaluation metrics.

Finally, the JL2P [1] code release does not include the evaluation. We therefore chose to reimplement our evaluation which we compute directly on 3D coordinates, rather than the standardized (mean subtraction, division by standard deviation) version as explained in Section 4.1 of the main paper.

Perceptual study details. The results of the pairwise comparisons in Figure 3 of the main paper are obtained as follows. We randomly sample 100 test descriptions and generate motion visualizations (as in the supplemental video [10]) from all the three previous methods [1, 4, 6], our method, and the ground truth (500 videos). From these visualizations, we create pairs of videos for each comparison, randomly swapping the left-right order of the video in each question (also 500 videos). For the semantic study, we display the text as well as the pair of videos simultaneously to Amazon Mechanical Turker (AMT) workers. The workers are asked to answer the question: "Which motion corresponds better to the textual description?". For the realism study, we use the same set of motion pairs and display them to other AMT workers but without showing the text description. They are asked to answer the question: "Which motion is more realistic?". For both studies, each worker answers a batch of questions, where the first 3 questions are discarded and used as a 'warmup' for the task. We further added 2 'catch trials' to detect unqualified workers, whose batch we discarded in our evaluation. We detected exactly 20 such workers out of 100 in both studies. Each pair of videos is shown to multiple workers between 2 and 5 (4 in average), from which we compute a majority vote to determine which generation is better than the other. If there is a tie, we assign a 0.5 equal score to both methods. The resulting percentage is computed over the 100 test descriptions.

For the semantic study, posing the question as a pairwise comparison can lead to a bias towards a more realistic motion (although the semantic correspondence to the text may be worse). To disentangle such realism bias in pairwise comparisons, we repeat the semantic perceptual study with one video at a time: workers on AMT were asked to rate how much they agree with the following statement: "*The body motion correctly represents the textual description*" by choosing between (1) *Strongly disagree*, (2) *Disagree*, (3) *Neither agree nor disagree*, (4) *Agree*, or (5) *Strongly agree*. Using a 20-sequence test set, we generated 4 batches with our method and 1 batch with Ghosh et al. [4]. At the beginning, we added 2 examples along with instructions to explain what we expect from 'correctness' and to disentangle from the realism: (i) a realistic motion (from ground truth) that does not correspond to the text, and (ii) a relatively less realistic motion (from generations) that does correspond to the text. We added 4 warm-up examples after them, and 3 catch trials at random locations. We ran the study with 24 AMT workers but half of them failed the catch trials (and the remaining results were not very consistent). So we repeated this experiment with naive lab members. Again we used all the examples at the beginning and catch trials. With this setup, we obtained 21 completed batches. The users rated an average of **3.50** correctness score out of 5 likert scale for our generations versus

3.04 for Ghosh et al. [4]. To further assess the quality of the diverse generations, we generated 5 samples per test description for TEMOS and obtained an average of **3.54** $\pm$ **0.1** score, showing that we preserve correctness within diverse generations as well.

D Implementation details

Architectural details. For all the encoders and the decoder of TEMOS, we set the embedding dimensionality to 256, the number of layers to 6, the number of heads in multi-head attention to 6, the dropout rate to 0.1, and the dimension of the intermediate feedforward network to 1024 in the Transformers.
Library credits. Our models are implemented with PyTorch [9] and PyTorch Lightning [3]. We use Hydra [17] to handle configurations. For the text models, we use the Transformers library [16].
Runtime. Training our TEMOS model takes about 4.5 hours for 1K epochs, with a batch size of 32, on a single Tesla V100 GPU (16GB) using about 15GB GPU memory for training (i.e., *16 seconds* per epoch). In comparison, according to their paper, Ghosh et al. [4] trained their model for 350 epochs on a single Tesla V100 GPU in about 15 hours (i.e., *154 seconds* per epoch). While the rest of the hardware specifications or implementation efficiency may vary, given the same type of GPU for both methods, we can expect that our model trains an order of magnitude faster. This may be because our model generates the full motion with only one decoder pass. Previous work produces one frame at a time iteratively (i.e., the next frame has to wait for the previous one to be generated).

E KIT Motion-Language text statistics

In the KIT Motion-Language [12] dataset, there are 3911 motions and a total of 6352 text sequences (in which 900 motions are not annotated). Using a natural language processing parser, we extract "action phrases" from each sentence, based on verbs. For example, given the sentence "A human walks slowly", we automatically detect and lemmatize the verb, and attach complements to it, such that it becomes "walk slowly". With this procedure, we group sequences that correspond to the same action phrase and detect 4153 such action clusters out of 6352 sequences. The distribution of these clusters is very unbalanced: "walk forward" appears 596 times while there are 4030 actions that appear less than 10 times (3226 of them appear only once). On average, an action phrase appears 2.25 times. This information shows that the calculation of distribution-based metrics, such as FID, is not relevant for this dataset.

Bibliography

[1] Ahuja, C., Morency, L.P.: Language2Pose: Natural language grounded pose forecasting. In: International Conference on 3D Vision (3DV) (2019) 4, 7, 8

[2] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: North American Chapter of the Association for Computational Linguistics (NAACL) (2019) 3, 4

[3] Falcon et al., W.: Pytorch lightning. GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning (2019) 9

[4] Ghosh, A., Cheema, N., Oguz, C., Theobalt, C., Slusallek, P.: Synthesis of compositional animations from textual descriptions. In: International Conference on Computer Vision (ICCV) (2021) 4, 7, 8, 9

[5] Holden, D., Saito, J., Komura, T.: A deep learning framework for character motion synthesis and editing. ACM Transactions on Graphics (TOG) (2016) 5

[6] Lin, A.S., Wu, L., Corona, R., Tai, K., Huang, Q., Mooney, R.J.: Generating animated videos of human activities from natural language descriptions. Visually Grounded Interaction and Language (ViGIL) NeurIPS Workshop (2018) 4, 8

[7] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692 (2019) 3, 4

[8] Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: SMPL: A skinned multi-person linear model. ACM Transactions on Graphics (TOG) (2015) 6

[9] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep learning library. In: Neural Information Processing Systems (NeurIPS) (2019) 9

[10] Petrovich, M., Black, M.J., Varol, G.: TEMOS project page: Generating diverse human motions from textual descriptions. `https://mathis.petrovich.fr/temos/` 1, 8

[11] Petrovich, M., Black, M.J., Varol, G.: Action-conditioned 3D human motion synthesis with transformer VAE. In: International Conference on Computer Vision (ICCV) (2021) 1

[12] Plappert, M., Mandery, C., Asfour, T.: The KIT motion-language dataset. Big Data (2016) 9

[13] Romero, J., Tzionas, D., Black, M.J.: Embodied hands: Modeling and capturing hands and bodies together. ACM Transactions on Graphics (TOG) (2017) 6

[14] Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019) 3, 4

[15] Terlemez, O., Ulbrich, S., Mandery, C., Do, M., Vahrenkamp, N., Asfour, T.: Master motor map (MMM) — framework and toolkit for capturing, representing, and reproducing human motion on humanoid robots. In: International Conference on Humanoid Robots (2014) 5

[16] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Empirical Methods in Natural Language Processing: System Demonstrations (2020) 9

[17] Yadan, O.: Hydra - a framework for elegantly configuring complex applications (2019), https://github.com/facebookresearch/hydra 9

[18] Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: Computer Vision and Pattern Recognition (CVPR) (2019) 6