PoseFix: Correcting 3D Human Poses with Natural Language
– *Supplementary Material* –

Ginger Delmas[1,2], Philippe Weinzaepfel[2], Francesc Moreno-Noguer[1], Grégory Rogez[2]

[1] Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain
[2] NAVER LABS Europe
[1]{name.surname}@naverlabs.com, [2]{gdelmas, fmoreno}@iri.upc.edu

In this supplementary material, we first provide additional details and statistics on the PoseFix dataset in Section A. The original triplets from PoseFix for the generated results presented in the main paper are available in Section B. Additional visualizations are provided in Section C. Finally, we give implementation details in Section D.

A. PoseFix complementary information

In this section, we provide additional details about the creation of the PoseFix dataset.

A.1. Human annotations

Sequences of origin. The poses in PoseFix were extracted from AMASS [5]. In Figure A1, we present the proportion of poses coming from each of the datasets included in AMASS. We notice that most poses belong to the DanceDB dataset (44%), presumably because this is where the poses are the most diverse. Recall that poses were chosen following a farther-point sampling algorithm to ensure we would get a various subset of poses. Besides, we note that most of the sequences available in DanceDB (94%) and MPI-limits (83%) provided at least one pose to PoseFix, which suggests that PoseFix could help in apprehending very complex, extreme poses.

Turkers qualifications and statistics. The annotations were collected on Amazon Mechanical Turk. Participating workers ("Turkers") had to come from English-speaking countries (Australia, Canada, New Zealand, United Kingdom, USA), have completed at least 5,000 other tasks, and have an approval rate greater than 95%. In total, 105 different annotators participated. We qualified 20 of them for access to the larger batches, on the basis of at least 3 good annotations. Other 50 workers were excluded from our annotation task because of poor writing, misunderstanding of the task or cheating. The remaining participants did not complete enough annotations of good quality to be qualified for accessing more. Eventually, over 90% of the annotations were made by 8 annotators.

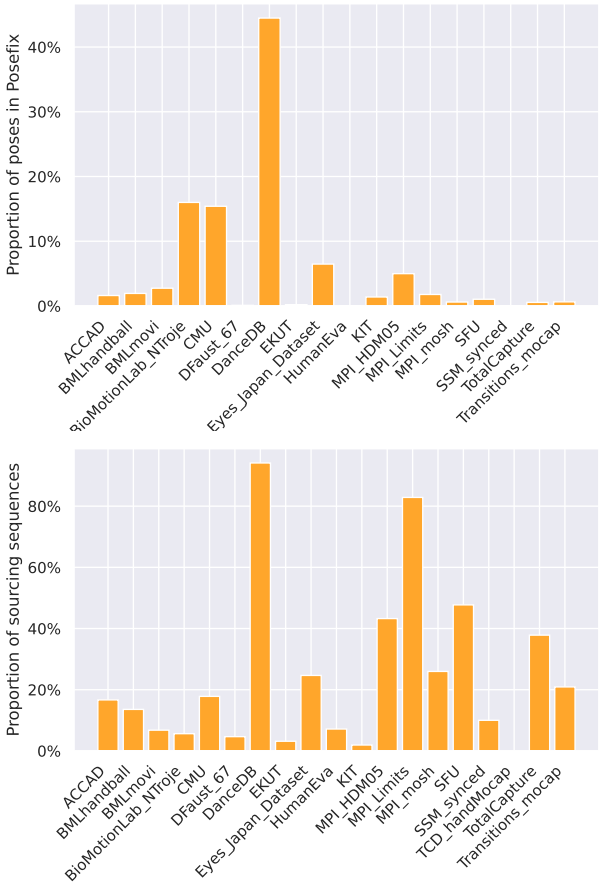

Figure A1. **Origin of the human-annotated poses in PoseFix.** The top plot shows the proportion of poses in PoseFix that come from each sub-dataset in AMASS [5]. The lower plot shows the proportion of sequences, in each of the sub-dataset, that provided at least one pose to PoseFix.

Pricing. Properly completing an annotation, after a bit of

training, was timed to take approximately 1'10". Annotations from the smaller qualifying batches were rewarded \$0.25. Once a worker completed 3 of them correctly, s/he was granted access to the larger batches, where annotations were rewarded \$0.32 each, based on the minimum wage in California for 2023. We additionally paid a 10% bonus for every 30 annotations.

Quality assessment. Annotations from the early smaller qualifying batches which were opened to any worker were systematically reviewed. In contrast, only up to 10% of the trusted worker annotations were randomly selected for manual review. The quality of the annotations was assessed based on the following criteria:

- *completeness*: most of the differences between pose A and pose B were addressed in the annotation;
- *direction accuracy*: the annotation explains how to go from pose A to pose B, and not the reverse;
- *left/right accuracy*: the words '*left*' and '*right*' were used in the body's frame of reference;
- *3D consideration*: the annotation fits the 3D information, no guess was taken on occluded body parts, or ambiguous postures;
- *no distance metric*: the annotation does not contain any distance metric (*e.g.*, '*one meter apart*'), which would not scale to bodies of different size;
- *writing quality*: correct grammar and formulation.

Length of the human-written annotations. Figure A2 shows the length distribution of the collected annotations. We here refer to the length as the number of words, excluding punctuation. While the annotations were constrained to be at least 10-word long, they tend to count about 30 words, suggesting that the differences between two similar poses A and B are both subtle and several.

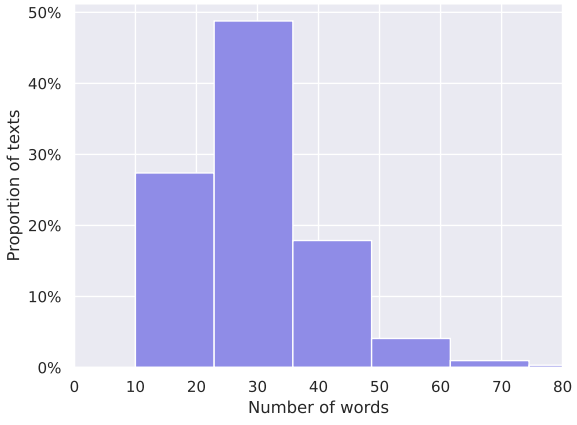

Figure A2. **Distribution of the number of words in the human-written annotations from PoseFix**.

A.2. Automatic annotations

We explain here in more details the learning-free process to automatically generate modifiers. The different steps of the pipeline are illustrated in Figure A3. We comment on some of those steps.

Code extraction. Two of the elementary paircodes are basically variation-versions of the initial posecodes [1]: we look at the change in angle posecode or distance posecode between pose A and pose B. The third kind of paircode studies the variation in position of a keypoint along the x-, y- or z- axis. All three paircodes are computed on the orientation-normalized bodies, so that the produced instructions would not depend on the change in global orientation of the body between pose A and pose B. This last part is treated separately, and yields a sentence that is added at the beginning of the modifier.
We also resort to the posecodes of both poses A and B to define super-paircodes, and thus gain in abstraction or formulation quality. There can be several ways to achieve the same paircode, each way comprising at least two conditions (posecode and paircode mixed together). Some posecodes of pose B, if statistically rare, are also included in the final modifier, *e.g.* '*the hands should be shoulder-width apart*', '*the left thigh should be parallel with the ground*'. Posecodes of pose A are only useful for super-paircode computations.

Code selection and aggregation. We proceed as in [1]. Trivial codes are removed. The codes (paircodes + posecodes) are aggregated based on simple syntactic rules depending on shared information between codes.

Code ordering. The final set of codes is semantically ordered to produce modifiers that are easier to read and closer to what a human would write (*i.e.*, describe about everything related to the right arm at once, instead of scattering pieces of information everywhere in the text). This step did not exist in the PoseScript automatic pipeline. Specifically, we design a directed graph where the nodes represent the body parts and the edges define a relation of inclusion or proximity between them (*e.g. torso→left shoulder*, *arm→forearm*). For each pose pair, we perform a randomized depth walk through the graph: starting from the *body* node, we choose one node at random among the ones directly accessible, then reiterate the process from that node until we reach a leaf; at that point, we come back to the last visited node leading to non-visited nodes and sample one child node at random. We use the order in which the body parts are visited to order the paircodes.

Code conversion. Codes are converted to pieces of text by plugging information into a randomly chosen template sentence associated to each of them. The pieces of text are next concatenated thanks to transition texts. Verbs are conjugated accordingly to the chosen transition (*e.g.* "while +

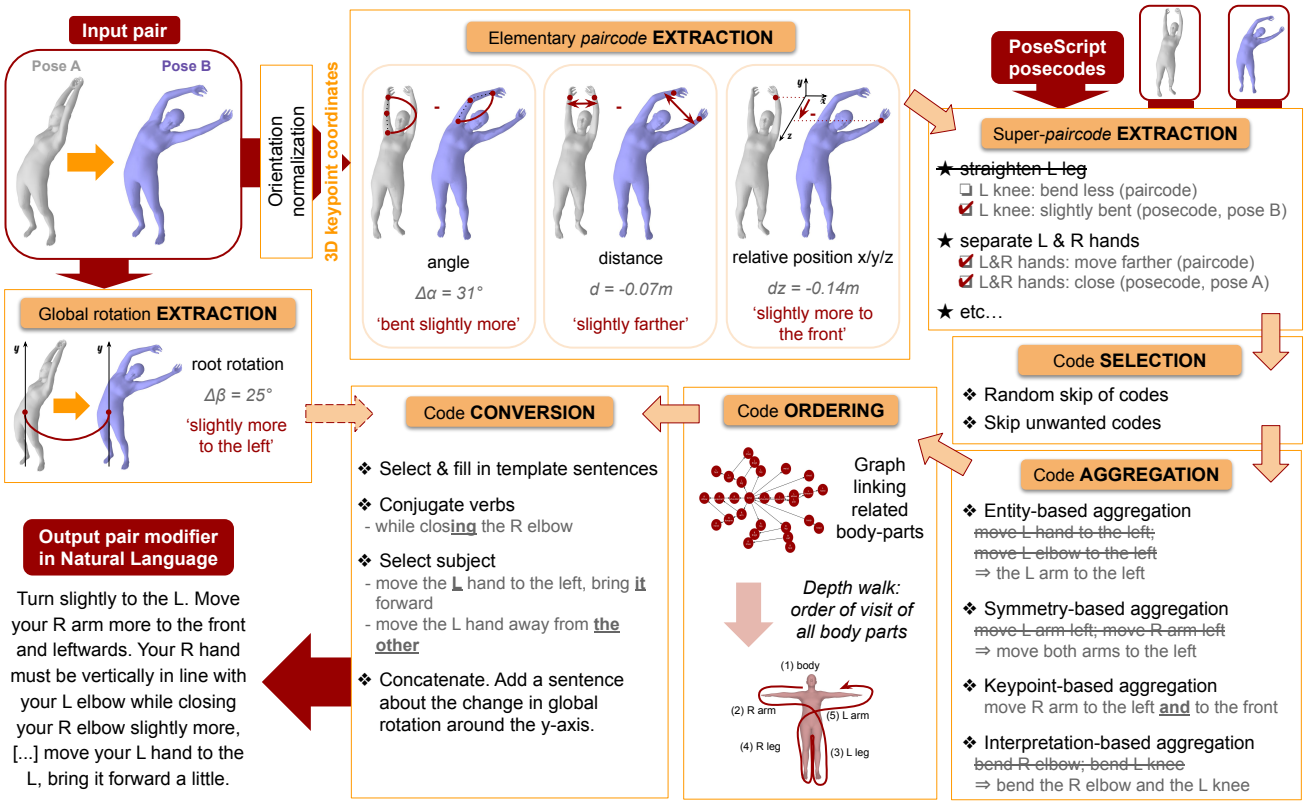

Figure A3. **Automatic Comparative Pipeline**, which generates modifiers based on the 3D keypoint coordinates of two input poses. L (resp. R) stands for 'left' (resp. 'right').

gerund") and code (*e.g.* posecodes lead to "[...] should be" sentences).

We refer to the code for the detailed and complete list of paircodes and super-paircodes definition.

B. Original triplet of the generation examples

In this section, we provide the original triplet for the generation results presented in Figure 5 of the main paper (see Figure A4) and in Figure 7 of the main paper (see Figure A5). While this ground truth may ease the comparison, it is not the only true answer for a generative model: multiple valid results could be produced. The GT was purposely omitted to prevent judgment bias, but is added here for reference.

C. Miscellaneous visualizations

Robot teaching application. The choice of modifiers in Natural Language to learn the difference between two poses proves especially useful in applications where direct manipulation is not possible, for instance in the case of robot teaching. Figure A6 shows a snapshot of a demo where a two-arm robot pose is optimized to match SMPL keypoints obtained from textual instructions.

The PoseCopy behavior. The PoseCopy setting for the text-based pose editing task consists in training the model with a proportion of the data where the text is emptied and pose B becomes a copy-paste of pose A. This training configuration makes it possible for the model to yield the exact same pose as the initial one, when no correctional instruction is specified, see Figure A7 for an example. Besides, we hypothesize that this setting encourages the model to better pay attention to pose A.

D. Implementation details

Architecture details. We follow the VPoser [6] architecture for our pose encoder, modified to account for the 52 joints of the SMPL-H [7] body model. In the 'glove+bigru' configuration of our pose editing baseline, GloVe word embeddings are of size 300 and we use a bidirectional GRU with one layer and hidden state features of size 512. In the transformer configuration, we use a frozen pretrained DistilBERT model to encode the text tokens. The transformer afterwards is composed of 4 layers with 4 heads and feed-forward networks with 1024 dimensions. It relies on GELU [2] activations and uses a 0.1 dropout. The text embedding is eventually obtained by performing an average pooling. The transformer in our correctional text genera-

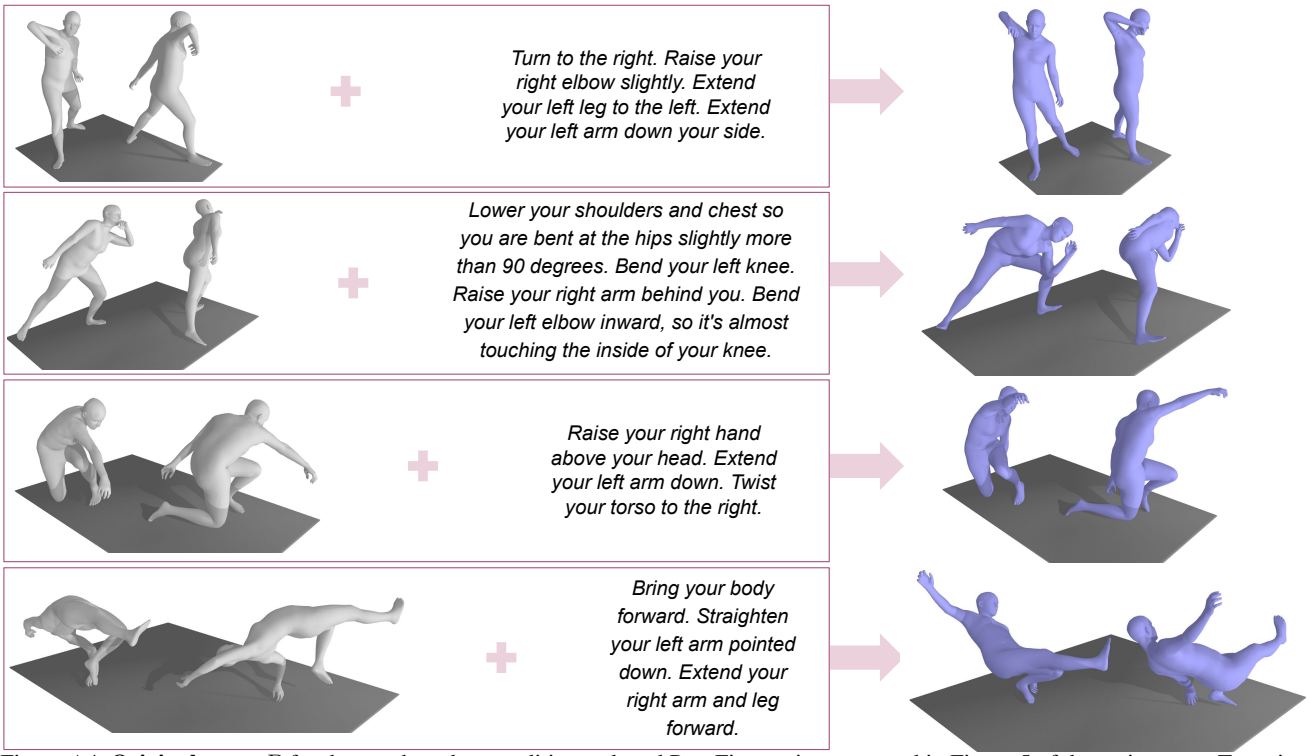

Figure A4. **Original poses** B for the text-based pose editing task and PoseFix queries presented in Figure 5 of the main paper. Two views of the each pose are shown on the same ground plane. Pose A is shown in grey, pose B in purple.

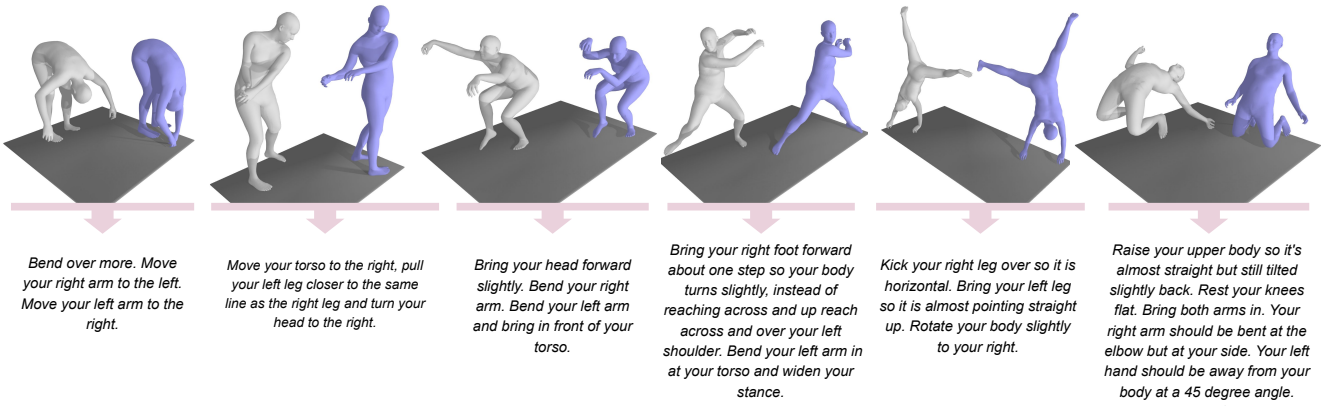

Figure A5. **Original correctional feedback annotation** for PoseFix pose pairs presented in Figure 7 of the main paper. Pose A is shown in grey, pose B in purple.

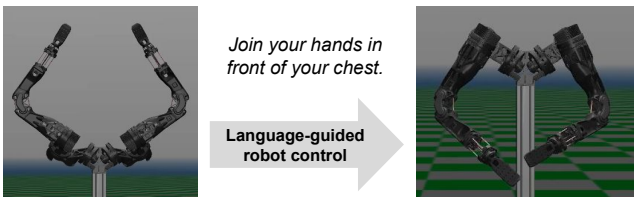

Figure A6. **Robot teaching application.**

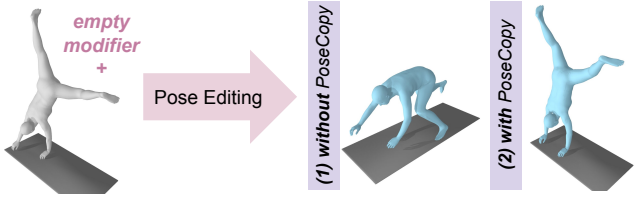

Figure A7. **Effect of training with PoseCopy.**

tion baseline is the same as for pose editing, except that we use 8 heads. In our models for both tasks, the poses and texts are encoded in latent spaces of dimensions d=32 and n=128 (n=512 for the text generation task) respectively.

Optimization and training details. We trained our models with the Adam [3] optimizer, a batch size of 128, a learning rate of 10^{-5} (10^{-4} for pretraining; and 10^{-6} for fine-

tuning in the case of pose editing) and a weight decay of 10^{-4} (10^{-5} for finetuning in the case of pose editing). The pose editing model was trained for 10,000 epochs (half for pretraining and half for finetuning, or 10,000 straight if no pretraining was involved), while the text generation model was trained for 3,000 epochs for pretraining and 2,000 for finetuning. In the *PoseCopy* setting, 50% of the batch is randomly used in "copy" mode (*i.e.*, empty text, with poses A and B being the same).

Why using the ELBO metric? The ELBO is well suited to VAEs [4]: it balances reconstruction and KL into a lower bound on the data log likelihood, a universal quantity for comparing likelihood-based generative models. It accounts for the probabilistic nature of the model, by evaluating the target under the output distribution. In a VAE framework, reporting reconstruction errors only does not penalize the model for storing a lot of information in the latent variable produced by the encoder. The extreme case of an encoder that learns an identity function would appear optimal, yet fail at test time when the ground truth is no longer available for encoding. By contrast, the ELBO takes both reconstruction and the amount of information given by the encoder (the KL term) into account, and combines them into a lower bound on the data log likelihood.

Hand data. We used the hand data (fingers joints) for all ours experiments, but note that this was not necessary, given that the hands all have the same pose for PoseFix human-annotated pose pairs. In case more data with relevant hand information is annotated in the future, we suggest to keep the original hand data for the pairs annotated in this version of the dataset, as some annotators may have referred to them in their instructions.

References

[1] Delmas, Ginger and Weinzaepfel, Philippe and Lucas, Thomas and Moreno-Noguer, Francesc and Rogez, Grégory. PoseScript: 3D Human Poses from Natural Language. In *ECCV*, 2022. 2

[2] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 3

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 4

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 5

[5] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. AMASS: Archive of motion capture as surface shapes. In *ICCV*, 2019. 1

[6] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3D hands, face, and body from a single image. In *CVPR*, 2019. 3

[7] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. In *SIGGRAPH Asia*, 2017. 3