Supplementary File for "Being Comes from Not-being: Open-vocabulary Text-to-Motion Generation with Wordless Training"

1. Model Structure and Training Details

Our OOHMG consists of two generators, i.e., the text-to-pose and motion generators. To optimize the text-to-pose generator, we also distill a text-pose alignment model, namely TPA, from the versatile CLIP [10]. To this end, these three neural networks contribute to our OOHMG in this paper. In this part, we describe the format of input and output as well as the architecture for these networks.

Fundamental neural network architectures. There are mainly two kinds of neural network architectures used in this paper, i.e., ResNet-based networks and Transformer-based networks. For **ResNet-based networks**, input poses are projected into embeddings by a linear layer, and then processed by 6 residual blocks. The intermediate results are normalized by a layer normalization layer and another linear layer to obtain the final results. The residual block will first normalize the input by a layer normalization, and then forward the normalized embeddings to a Linear-GELU-Dropout(0.1)-Linear-Dropout(0.1) networks to predict the residual which will be added to the normalized embeddings to form the output of the residual block. The hidden size is 1024. As for **Transformer-based networks**, we adopt a similar architecture as Bert [1]. The architectures of the transformer encoder and decoder layer are implemented by PyTorch [5]. The poses of a motion are first projected by a linear projection layer, then processed by an 8-layered transformer encoder/decoder, and finally fed to an estimation layer to obtain a prediction. The number of attention heads is 8, the hidden size is 1024 and the dropout rate is 0.1.

TPA. TPA is distilled from CLIP for aligning 3D poses and texts. Specifically, for the text encoder, TPA simply reuses the text encoder of CLIP. As for the pose encoder, TPA adopts the ResNet-based network. TPA pose encoder takes in the 6D-rotation representation of the pose and predicts the output of the original pipeline. The batch size is 1024, and the learning rate is 1e-4 at the beginning and annealed by the CosineAnnealingLR scheduler implemented by PyTorch. The number of training iterations is 1e6. The training curves of loss in learning rates are plotted at the left of Fig. 1. As for the process of the **original pipeline**, we mostly adopt the process used in Avatar-CLIP [2]. Specifically, as shown in our manuscript, the 3D pose representation is first used to generate the 3D meshes by SMPL [6, 8]. Then, 5 look-at cameras, with azimuth angles $[120, 150, 180, 210, 240]$ and fixed elevation, render the mesh into 5 images. After that, the image encoder of CLIP extracts the features of images and the pipeline takes the average for the features as the features of the 3D pose. The training poses are sampled from AMASS [3].

Table 1. Ablation results for our text-to-pose generator with different λ_{L2} evaluated on BABEL [9].

	CLIP Score ↑	In-distrib. ↓	Top50 ↑
$\lambda_{L2} = 0$	**0.2711**	0.0111	**0.7224**
$\lambda_{L2} = 0.05$	0.2702	0.0019	0.7039
$\lambda_{L2} = 0.1$	0.2694	0.0015	0.6711
$\lambda_{L2} = 0.15$	0.2689	**0.0012**	0.6446

Text-to-pose generator. The architecture of our text-to-pose generator is the ResNet-based network. It takes the text features extracted by TPA/CLIP text encoder and predicts the latent pose of the VPoser which is decoded by the VPoser decoder to obtain the 6D-rotation pose representation. During training, the 6D-rotation representation is fed to the TPA pose encoder for the pose feature. The batch size is 1024, and the learning rate is 1e-3 at the beginning and annealed by the CosineAnnealingLR scheduler implemented by PyTorch. The number of training epochs is 1K and the number of iterations of each epoch is 1K. As for selection for λ_{L2}, we found that when λ_{L2} equals 0.1, the performances of different metrics are more in balance as shown in Tab. 1. The loss curves with different λ_{L2} are plotted in the right of Fig. 1. As for noise features, the noise features are randomly sampled either from Normal distribution $\mathcal{N}(0; 1)$ or from Uniform distribution $\mathcal{U}[-1, 1]$. The proportions of the features from these two distributions are 50% and 50%. In addition, a random bias sampled from $\mathcal{U}[-1, 1]$ is added to each of the noise features.

Pretrained motion generator. As described in the main text, our text-to-motion generation uses the combination of a pretrained motion model and pose prompt. And the pretrained motion model is the only network in this stage.

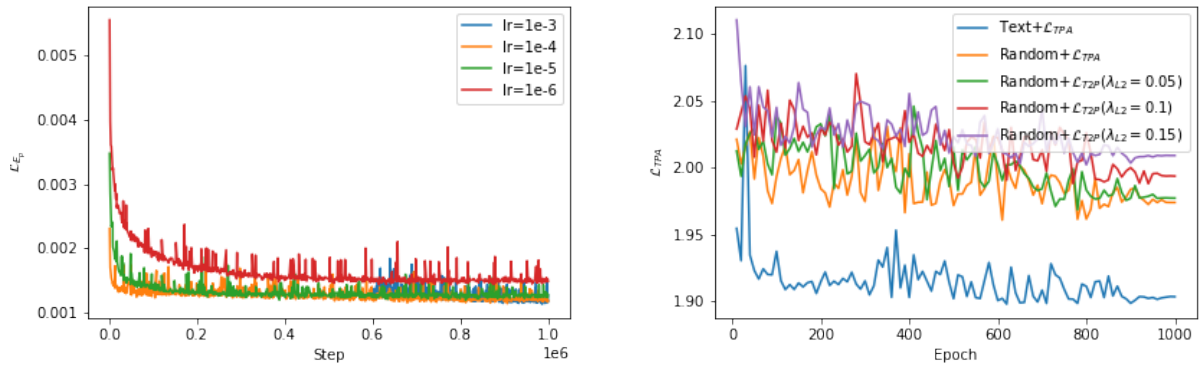

Figure 1. Training curves of our TPA and text-to-pose generator. The left figure includes the loss curves of TPA with different learning rates. The right figure includes the $\mathcal{L}_{\text{TPA}}$ curves of the text-to-pose generator with different training loss functions.

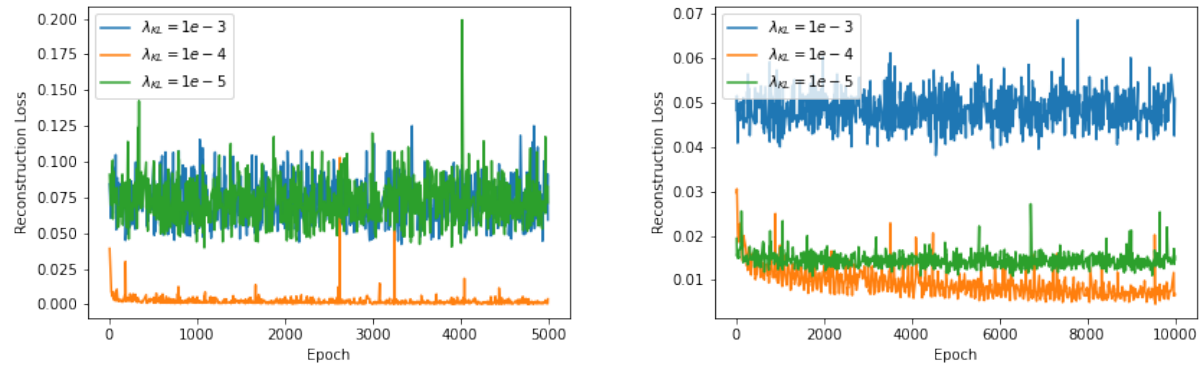

Figure 2. Training curves of our motion generator and general motion VAE which is used for evaluation. The left figure ablates the λ_{KL} of our motion generator; The right figure ablates the λ_{KL} of the motion VAE which is used for evaluation.

The pretrained motion model uses the transformer-based network architecture. As described in our manuscript, the pretrained motion model includes a motion encoder and a motion generator. During the training phase, the motion encoder takes in a motion with 6D-rotation representations and two tokens for mean and standard deviation. The predicted mean and standard of the encoder are used to sample latent code via the reparameterization trick. The motion generator takes the latent code and randomly masks motion as input to reconstruct the complete motion. The batch size is 64, and the learning rate is 1e-4 at the beginning and annealed by the CosineAnnealingLR scheduler implemented by PyTorch. The number of training epochs is 5K. λ_{KL} is set as 1e-4 empirically. The reconstruction loss for different λ_{KL} is shown at the left of Fig. 2. During inference, the values of the latent code are set as 0.

2. Experiment Details

2.1. Baselines Details.

The results of all baseline methods are obtained by running their open-released codes. As for MotionCLIP [11], we directly adopt their open-released model for motion generation. As for Interpolation / Matching / Optimize / VPoserOptimize / AvatarCLIP [2], we adopt and use their open-released code and make small revisions for evaluations. Particularly, Interpolation and AvatarCLIP are originally developed for generating motion using Top5 poses of a text. Therefore, these methods are designed to use a fixed number of condition poses with similar semantics. In our experiments, to evaluate the controllability, the semantics and number of condition poses are different. Thus, we adapt the original code to evaluate the controllability. In the other experiment we simply their original code for evaluation.

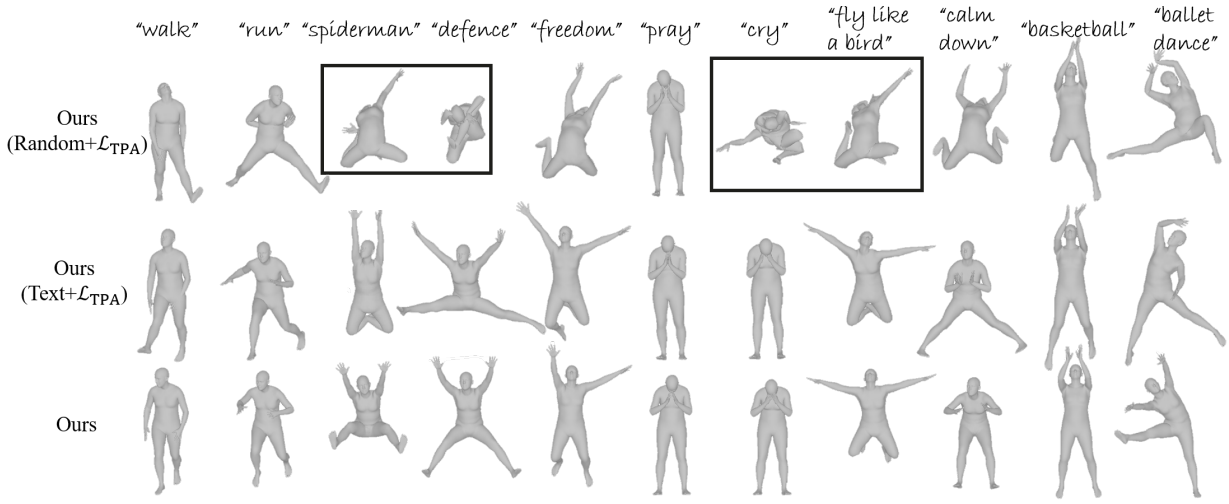

Figure 3. The visual results of different combinations of wordless training and real-world text supervision.

Table 2. CLIP scores of our text-to-pose generator trained with maximizing TPA score or $\mathcal{L}_{\text{TPA}}$ using the checkpoints of TPA at iteration 1e4, 1e5 and 1e6.

Iterations	1e4	1e5	1e6
Text+Score	0.2491	0.2524	0.2620
Text+$\mathcal{L}_{\text{TPA}}$	0.2557	0.2613	0.2601

Table 3. Comprehensive results with/without adding an initial pose to the prompt. The arrow ↑ indicates the performance is better if the value is higher.

	In-distrib.↓	Top1↑	Top10↑	Top50↑
Ours wo init. pose	0.0208	0.0768	0.3135	0.6154
Ours	**0.0205**	**0.0792**	**0.3231**	**0.6494**

2.2. OOHMG Prompt Details

Given a text, our text-to-pose generator synthesizes the text-consistent pose and places it in the middle of a sequence of masks to form the prompt. However, we found that in this manner, the motion generator usually generates a motion filled with similar poses. This is reasonable since the motion filled with similar poses might also exist in the real world. Thanks to the strong controllability of the motion generator, we can easily adjust the generated motion by refining the prompts with multiple poses. For example, we can use descriptions to specify the fore-pose, middle pose, and post-pose to construct a motion prompt. In our experiments, we found that adding an initial pose (which latent poses of VPoser is a zero vector) to the prompt can significantly improve the variation of the generated motions. And we also found that in this manner, the performance of motion evaluation also improve a little bit, as presented in Tab. 3.

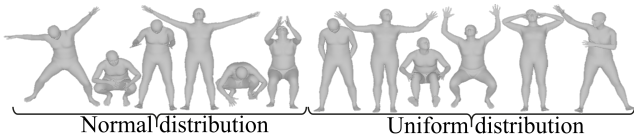

Figure 4. The visualization of poses generated with random text features.

2.3. Wordless Training Visualization

The poses in Fig. 3 are generated w.o. / w. wordless training, corresponding to "Ours(Text+$\mathcal{L}_{\text{TPA}}$) / Ours(Random+$\mathcal{L}_{\text{TPA}}$)" in Tab.4 of the main paper. We observe that, without wordless training, the generator may not perform well with some unseen texts (in boxes).

2.4. Visualization for poses generated with random text features

In Fig. 4, we show several poses generated by our text-to-pose generator using random features obeying Equ.(5) in the main paper. The results imply that our method can generalize to random texts.

2.5. Contrastive Loss and Maximizing Score

As we can see in Tab. 2, when TPA is finished training at iter 1e6, by maximizing the TPA score (i.e., Text+Score) can also obtain comparable performance to Text+$\mathcal{L}_{\text{TPA}}$ which uses $\mathcal{L}_{\text{TPA}}$ for optimization. However, if TPA is not converged, Text+Score is more likely to have a degenerated performance. While Text++$\mathcal{L}_{\text{TPA}}$ has more stable performance. We contribute such stability to more dense supervision by drawing other samples into the contrastive loss.

2.6. Evaluation Metrics.

In our experiments, we mainly evaluate text-to-pose and text-to-motion generations. Unfortunately, there are no suitable evaluation metrics in the current literature. In AvatarCLIP [2], they only conduct user studies. Therefore, our paper proposes to adopt and adapt popular metrics to evaluate performance.

As for text-to-pose generation, we mainly evaluate the CLIP similarity score (i.e., **CLIP Score**), in-distribution distance (i.e., **In-distrib.**), text-to-pose-to-text reconstruction loss (i.e. **Cycle loss**) and CLIP-R-precision [4] (i.e., **TopK**). Specifically, In-distrib. is the reconstruction loss of the VPoser [6]. If the generated pose is similar to the training poses of VPoser, the reconstruction is likely to be small. And for Cyc. Loss, we train an auxiliary neural network for each method to learn the reverse mapping from the generated poses to their corresponding text features. And if the regression loss is smaller, the generated poses are more likely to carry more textual information and thus more diverse. The structure of the auxiliary regression model is similar to the pose encoder of TPA. The hidden size is 512 and the number of ResBlock is 2, and no dropout. The learning rate is 1e-3 and the number of iterations is 1e4.

As for text-to-motion generation, we mainly evaluate the in-distribution degree (i.e., **In-distrib.**) and the extended CLIP-R-precision for motion (i.e., **TopK**). Specifically, In-distrib. also uses reconstruction loss of a pretrained motion VAE. The pretrained motion VAE is similar to the architecture of ACTOR [7] without condition input. The KL loss term is 1e-4 as ablated in the right of Fig. 2. As for the extended CLIP-R-precision, we say that the text-motion matching is accurate if, among all poses of the generated motion from different texts, the best-matched pose of the text is located in the generated motion of the text. To achieve a better motion-level CLIP-R-precision, the generated motion should 1) contain the text-consistent poses, and 2) and does not contain irrelevant poses that might cause mismatching for other poses.

To measure whether the motion generator can synthesize motion according to the given poses, we also introduce the KP metric. We randomly sample poses from 4096 clustered poses from AMASS and use them as conditional poses. And we measure whether the generated motion contains that pose by calculating the minimal reconstruction error of these poses. The small the reconstruction error is, the better the generated motion preserves the conditional poses. We use the 4096 clustered poses used in AvatarCLIP [2] as the condition poses. We randomly sample from the clustered poses to construct the *K*P test set, where $K \in \{1, 2, 3\}$ indicates the number of the condition poses used for generation a motion. The measurement of *K*P for a generated motion m conditioned on poses $\{p'_k\}_{j=1:K}$ is formulated as:

$$K\text{P}(m, \{p'_k\}_{j=1:K}) = \frac{1}{K} \sum_{k=1:K} \min_{p_j \in m} ||p'_k - p_j||_2. \quad (1)$$

2.7. Human Evaluation

For human evaluation, we designed our human evaluation questionnaires in the free online platform (https://wj.qq.com/). We shared the questionnaires on the internet with the non-paid and unknown subjects who are not participated in our work, including but not limited to colleagues from different universities, workers from different industrial companies, etc. For each question in the questionnaire, a subject will be provided with a text description and several shuffled generation contents from different methods, following two queries in terms of text consistency and realness. We randomly invite 25 human evaluators to compare the performance of pose generation and motion generation of different methods. For each participant, we inquire about 50 questions (25 for pose and 25 for motion). A text and the generated poses/motions of different methods are given for each question. The participants are required to give the order of methods in terms of realness and text consistency. For realness, we ask the participant which pose/motion is more vivid as real-world pose/motion. And for text consistency, we ask them which pose/motion is more in line with the given text. To avoid the participant trivially giving the meaningless order, we randomly change the order of the presentation order of different methods. There are three pose generation methods and four motion generation methods. For the pose generation method with ranks 1st, 2nd, and 3rd, we assign scores 3, 2 ad 1 for each question. Similarly, for the motion generation method with ranks 1st, 2nd, 3rd, and 4th, we assign scores 4, 3, 2 ad 1 for each question. To better understand the content of the human evaluation, we also include the visualization of poses and motions of different methods, which are placed in a separate folder along with the supplementary.

2.8. Discussion and Limitations

As foundation models, e.g., CLIP, become more mature and learn more real-world knowledge, it provides us with new opportunities and challenges to a new learning paradigm. In this paper, we show one of the possibilities that learning from the foundation model instead of learning from data. We believe such attempts have an advantage over learning from data since the foundation model can better associate multi-modality data to make better decisions. Particularly, in our method, we found that using noisy training data can probe diverse knowledge out of the foundation model, which implies the feasibility of building an agent that can actively and continuously learn knowledge from

the foundation model starting from chaos, i.e., noises, without manually feeding data which might limit the learnable knowledge of the foundation model. By this means, the agent might be able to learn something that is existed but we have not thought of yet or tasks we cannot formulate mathematically using our current knowledge.

Although our method is mainly offline generation, our method can also be extended to online generation. In addition to pure online generation, ours may provide a better initial solution to speed up the optimization and improve robustness.

However, as one of the few pioneers, several aspects can be improved in our work. One is that CLIP learns from static image data and cannot handle motion description. It cannot handle some difficult texts like a sentence having multiple successive motions. However, this can be addressed by a divide-and-conquer strategy. And with the great controllability of our method, our methods can be easily extended to handle this problem. And our work mainly evaluates the existing methods using CLIP-based measurements, e.g., CLIP-R-precision, since the compared methods are mostly CLIP-based. Nevertheless, there are several foundation models for aligning video and texts, but we found that most of them are learning with limited types of video data and are not as general as CLIP due to the difficulty of data collection for video training data. To this end, in our paper, we still prefer CLIP for zero-shot learning. And we leave the research with other foundation models in the future.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[2] Fangzhou Hong, Mingyuan Zhang, Liang Pan, Zhongang Cai, Lei Yang, and Ziwei Liu. Avatarclip: Zero-shot text-driven generation and animation of 3d avatars. *arXiv preprint arXiv:2205.08535*, 2022. 1, 2, 4

[3] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. Amass: Archive of motion capture as surface shapes. *international conference on computer vision*, 2019. 1

[4] Dong Huk Park, Samaneh Azadi, Xihui Liu, Trevor Darrell, and Anna Rohrbach. Benchmark for compositional text-to-image synthesis. *neural information processing systems*, 2021. 4

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *neural information processing systems*, 2019. 1

[6] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 4

[7] Mathis Petrovich, Michael J. Black, and Gül Varol. Action-conditioned 3d human motion synthesis with transformer vae. *international conference on computer vision*, 2021. 4

[8] Leonid Pishchulin, Stefanie Wuhrer, Thomas Helten, Christian Theobalt, and Bernt Schiele. Building statistical shape spaces for 3d human modeling. *Pattern Recognition*, 2017. 1

[9] Abhinanda R. Punnakkal, Arjun Chandrasekaran, Nikos Athanasiou, Alejandra Quiros-Ramirez, and Michael J. Black. Babel: Bodies, action and behavior with english labels. *computer vision and pattern recognition*, 2021. 1

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *international conference on machine learning*, 2021. 1

[11] Guy Tevet, Brian Gordon, Amir Hertz, Amit H Bermano, and Daniel Cohen-Or. Motionclip: Exposing human motion generation to clip space. *arXiv preprint arXiv:2203.08063*, 2022. 2