OhMG: Zero-shot Open-vocabulary Human Motion Generation

Junfan Lin, Jianlong Chang, Lingbo Liu, Guanbin Li, Liang Lin, Qi Tian, Chang-wen Chen [*] [†]

Abstract

Generating motion in line with text has attracted increasing attention nowadays. However, open-vocabulary human motion generation still remains touchless and undergoes the lack of diverse labeled data. The good news is that, recent studies of large multi-model foundation models (e.g., CLIP) have demonstrated superior performance on few/zero-shot image-text alignment, largely reducing the need for manually labeled data. In this paper, we take advantage of CLIP for open-vocabulary 3D human motion generation in a zero-shot manner. Specifically, our model is composed of two stages, i.e., text2pose and pose2motion. For text2pose, to address the difficulty of optimization with direct supervision from CLIP, we propose to carve the versatile CLIP model into a slimmer but more specific model for aligning 3D poses and texts, via a novel pipeline distillation strategy. Optimizing with the distilled 3D pose-text model, we manage to concretize the text-pose knowledge of CLIP into a text2pose generator effectively and efficiently. As for pose2motion, drawing inspiration from the advanced language model, we pretrain a transformer-based motion model, which makes up for the lack of motion dynamics of CLIP. After that, by formulating the generated poses from the text2pose stage as prompts, the motion generator can generate motions referring to the poses in a controllable and flexible manner. Our method is validated against advanced baselines and obtains sharp improvements. The code will be released here.

1 Introduction

Motion generation has attracted increasing attention due to its practical value in the fields of virtual reality, video games, and movies. As the motion capture techniques mature, motion data can be collected with fewer human efforts (Mahmood et al., 2019). However, when it comes to labeling the collected motions, the diversity of textual descriptions is usually affected by the labeling instructions and the knowledge of the crowd annotators, which might introduce unexpected selection biases (Pearl & Mackenzie, 2018) and limit the generality of the labeled data. The scarcity of diverse textual descriptions for different motions is one of the major obstacles to open-vocabulary motion generation. Nevertheless, the recent works on large-scale multi-model foundation models, e.g., CLIP (Radford et al., 2021b) or ALIGN (Jia et al., 2021), have shown the surprising capability to align diverse text and images in a few/zero-shot manner and largely reduce the need for manually labeling. Equipped with the foundation models, classical methods for text-to-image generation (Gal et al., 2021; Frans et al., 2021; Ramesh et al., 2022) and robotic vision grasping (Shridhar et al., 2022), are able to generalize to unseen textual descriptions or objects during inference. However, these methods stay at the level of using the feature representation ability of the foundation model. In this work, we take a step further and exploit the knowledge of the well-known foundation model, i.e., CLIP, to facilitate the zero-shot open-vocabulary 3D human motion generation.

The foundation model, CLIP (Radford et al., 2021b), is a language-image pretrained model for aligning images and texts. However, since it is only trained with static images, it implies that CLIP lacks the knowledge of motion dynamics for motion generation and can only be used for static

[*]linjf8@mail2.sysu.edu.cn, jianlong.chang@huawei.com, lingbo.liu@polyu.edu.hk, liguanbin@mail.sysu.edu.cn, linliang@ieee.org, tian.qi1@huawei.com, changwen.chen@polyu.edu.hk

[†]We are still working on improving this work. Therefore the experiment results might be different. The code for reproducing the experiment results will be released in the final version.

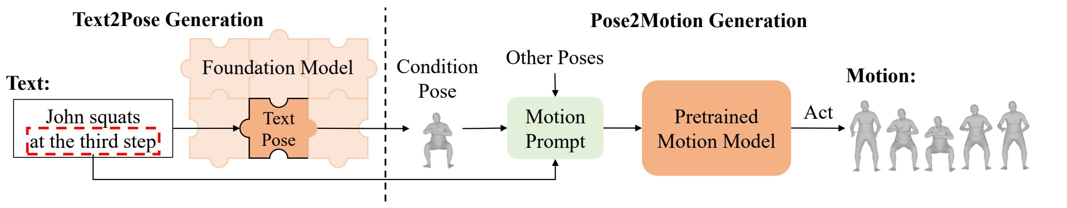

Figure 1: An overall sketch of OhMG. A text is fed to the text2pose generator to obtain a signature pose at the text2pose stage. Then, the pose can combine with other information, e.g., the text and the other poses, to create a motion prompt for the pretrained motion model to generate a motion.

pose generation. Therefore, to generate motions with CLIP, it's reasonable to leverage CLIP for static pose generation and then combine it with the motion dynamics learned from the collected motion data to generate the complete motion (Hong et al., 2022). In this paper, we adopt a two-stage text2motion generation model for zero-shot **O**pen-vocabulary **h**uman **M**otion **G**eneration, termed as OhMG. OhMG consists of two stages, i.e., text2pose and pose2motion stages. Briefly, text2pose translates a textual motion description into the signature pose of the motion, which is then served as a condition input to the pose2motion stage to obtain the complete motion. The overall sketch of OhMG is illustrated in Fig. 1. In the following, we further elaborate on each stage of our method.

At the text2pose stage, a text2pose generator learns to generate text-consistent 3D poses under the supervision of CLIP. However, as mentioned above, CLIP is not pretrained for aligning texts and 3D poses. To make use of CLIP, a promising workaround is to first generate the 3D human model with respect to the input 3D pose parameters, then project the 3D human model to 2D images and then apply CLIP to measure the alignment between the images and the input text (Hong et al., 2022). Ideally, when the whole pipeline is implemented in a differentiable manner, the input pose parameters could be adjusted and fit the input text via gradient descent. Unfortunately, in practice, we found it difficult to optimize the poses with this pipeline directly. The potential reason could be that, by using CLIP, there could be various images to present the same text (Ramesh et al., 2022; Zhou et al., 2022). And the rendered 2D images of the 3D human model might not be the optimal ones. In this case, optimization with CLIP might adjust the 3D pose in an unexpected direction. Furthermore, the pipeline is complex and requires substantial computing resources, which also hampers the optimization process. To address these problems, we propose a novel knowledge distillation method, termed pipeline distillation, which learns an end-to-end neural network to replace the complex pipeline. By learning from substantial poses and their output of the pipeline, pipeline distillation carves out a smaller but more specific model of CLIP for aligning 3D poses and texts. In our experiments, by using the distilled model, optimization becomes significantly efficient and we manage to learn a text2pose generator for generating open-vocabulary poses. Interestingly, we also found that the text2pose generator can be trained with noise sampled from Gaussian/Uniform distribution and be able to handle real-world text during inference. By doing so, the training process is not affected by the amount or types of the training texts resulting in a more generalized text2pose generator.

As for the stage of pose2motion, a motion generator is required to synthesize the motion containing the condition poses. Previous work (Hong et al., 2022) adopts the decoder of a pretrained motion VAE and searches in its latent space for preferable motion. The latent code is updated via gradient descent to minimize the distance between the poses of the decoded motion and the condition poses. However, this method requires iterative updates during deployment, which is time-consuming. Further, since the latent space of the motion VAE is high-dimensional and not necessarily convex, the generated motion is difficult to optimize. Differently, we view the relationship between poses and motion as the relationship between words and sentences in the field of natural language processing. And we pretrain a motion model via mask-and-reconstruction self-supervised learning as used in the advanced language model (Devlin et al., 2018). And during inference, the condition poses can be treated as the unmasked poses to prompt the motion model (Brown et al., 2020; Han et al., 2021) to synthesize the rest of the poses, resulting in a complete motion. We find that the motion model is easy and flexible to control for generating diverse motions given the condition poses.

Overall, the contributions of our OhMG are as follows. **For text2pose stage:** 1) We propose a novel text2pose generator that mines the knowledge from the multi-model foundation model, i.e., CLIP. 2) To overcome the difficulty of optimization, we propose a pipeline distillation strategy that turns the complex pipeline into a slimmer model for aligning 3D poses and texts. **As for pose2motion stage:** 3) We consider the motion generation as the same as the language modeling and pretrain a motion model via mask-and-reconstruction self-supervised learning. 4) Inspired by prompt design for probing knowledge from pretrained model (Han et al., 2021), we reformulate the condition poses as prompts to leverage the motion model to generate motion in a controllable and flexible manner.

2 RELATED WORK

Conditional Motion Generation. For conditional motion generation, there are various categories of research based on different types of conditions. In works (Aggarwal & Parikh, 2021), researchers use music as a condition to generate dance motions. While DVGANs (Lin & Amer, 2018), Language2Pose (Ahuja & Morency, 2019) and Text2Action (Ahn et al., 2018) synthesize motions according to short motion descriptions. And Action2Motion (Guo et al., 2020) and Actor (Petrovich et al., 2021) takes in action labels to generation motions. The success of these methods are heavily relied on the substantial motion capture datasets (Cai et al., 2022; 2021; Ionescu et al., 2014; Mahmood et al., 2019; Mehta et al., 2016; Varol et al., 2017; von Marcard et al., 2018), which are in forms of 3D keypoints or parameters of 3D human model SMPL (Pishchulin et al., 2017) as well as the labeled motion descriptions datasets, such as AMASS (Punnakkal et al., 2021), KIT motion-language dataset MatthiasPlappert2016TheKM, and HumanML3D dataset Guo et al. (2022). However, these datasets are limited by the task design and the difficulty of data collection. For example, the motions for emotion are not considered in these datasets. Learning with the limited datasets, although the methods Athanasiou et al. (2022); Zhang et al. (2022) have achieved qualitatively and quantitatively plausible results, they cannot generalize to open-vocabulary motion descriptions.

CLIP-based Zero-shot Text-driven Generation. Thanks to the fast development of the foundation models, zero-shot/few-shot learning is shown to have the potential to exceed the supervised learning (Radford et al., 2021a; Devlin et al., 2018; Brown et al., 2020). A recent text-image foundation model, CLIP (Radford et al., 2021a), which trains over a hundred million images gains a powerful visual and textual representation ability. It is shown to facilitate the classical visual tasks (Vinker et al., 2022), simply by providing a well-behaved, semantically structured, latent space. Combined with CLIP, DALL-E (Radford et al., 2021b) shows the amazing text2image synthesis capability by training with tremendous training data with CLIP features. Leveraging the strong generalization ability of CLIP, there are more and more zero-shot text-driven applications (Frans et al., 2021; Patashnik et al., 2021; Peng et al., 2021b). Using CLIP for tasks like NeRF or mesh, there are also a lot of progress in zero-shot text-guided generation (Jain et al., 2022; Jetchev, 2022; Michel et al., 2022; Sanghi et al., 2021; Peng et al., 2021a).

Related to ours, recent interesting studies combine CLIP with diffusion generation models (Rombach et al., 2022; Saharia et al., 2022) to generate text-consistent 3D meshes (Poole et al., 2022; Jain et al., 2021). Differently, these methods focus on generating static mesh, similar to static text-to-image generation. Close to motion sequence, recent works like Make-A-Video (Singer et al., 2022), Imagen Video (Ho et al., 2022) and Phenaki (Villegas et al., 2022) also extend text-image generation to text-video generation. Nevertheless, these methods are generating 2D images instead of intractable 3D models and motions. As for open-vocabulary motion generation, CLIP-Actor (Youwang et al., 2022) samples existing motion by matching the test textual descriptions with the motions labels of existing text-motion dataset. And MotionCLIP (Tevet et al., 2022) learns a motion VAE by regularized the latent space to align with the feature space of CLIP, which also requires labeled data. To the best of our knowledge, there is only one work, i.e., AvatarCLIP (Hong et al., 2022) that has the same setting as ours, which is also the first work to exploit the potential of zero-shot open-vocabulary human motion generation.

3 PRELIMINARIES

In this paper, we investigate the zero-shot open-vocabulary human motion generation by mining the text-pose knowledge from the CLIP (Radford et al., 2021b). In the following, we briefly introduce the task and the frequently-used notations as well as the CLIP model.

Task description and notations. Open-vocabulary 3D human motion generation takes in a natural language motion description d (for example, "Amy is shooting a basketball") and generates a motion

m in line with d. A motion is a sequence of 3D poses, $m = [p_t]_{t=1:T}$, where p is the 3D pose, t stands for the timestep and T is the maximum length of a motion. The representation of 3D pose p can be in different formats, e.g., an axis-angle representation $p^a \in \mathbb{R}^{J\times 3}$, an 6D-rotation representation $p^r \in \mathbb{R}^{J\times 6}$ (Zhou et al., 2019) or an compact latent representation $p^l \in \mathbb{R}^{32}$ of VPoser (Pavlakos et al., 2019) (a well-known pose VAE trained with massive poses). Here, J denotes the number of used joints in the human model and VPoser is a popular pretrained pose VAE model. In this paper, we use the 6D-rotation representation as the default representation. Following the setting in AvatarCLIP (Hong et al., 2022), we do not take the position transition, the facial expression, hand poses, and the global orientation into account. Therefore, J is 21 in this setting. Any of these representations can be used to generate 3D human meshes. In this paper, we use a popular parametric human model, SMPL, for its strong interpretability and compatibility with modern graphics platforms. SMPL (Loper et al., 2015) is a parametric human model driven by large-scale aligned human surface scans (Pishchulin et al., 2017). We can feed the pose parameters to SMPL to obtain 3D meshes v, denoted as $v = \mathcal{M}_{\text{SMPL}}(p)$. Meshes v are represented by a set of 3D positions of the vertices. Note that, the SMPL model also requires other input, such as faces and body shapes, which are set as default and ignored for brevity in our paper.

CLIP. CLIP (Radford et al., 2021b) is a vision-language pre-trained model trained with large-scale image-text datasets. It consists of an image encoder E_o and a text encoder E_d. Here, we use o to denote the image and d to represent the text. The encoders are trained in the way that the latent codes of paired images and texts are pulled together and the unpaired are pushed apart. Formally, the CLIP loss function is defined as

$$\mathcal{L}_{\text{CLIP}}(\mathbf{o}, \mathbf{d}) = -\sum_{i=1:B} \log \mathbf{Pr}(o_i|d_i) - \sum_{i=1:B} \log \mathbf{Pr}(d_i|o_i), \tag{1}$$

where $\mathbf{o}$ and $\mathbf{d}$ is the sets of images $\{o_i\}_{i=1:B}$ and texts $\{d_i\}_{i=1:B}$, and B is the batch size. **Pr** is the softmax probability of the o_i given d_i in a batch, vice versus. Particularly, to calculate $\mathbf{Pr}(o_j|d_i)$, the cosine similarity between text feature $f_{o_i} = E_o(o_i)$ and each image feature $f_{d_j} = E_d(d_j)$ of the batch data are calculated, and the temperature-softmax operation is applied to the cosine similarities. Formally, to calculate $\mathbf{Pr}(o_i|d_i)$:

$$\text{cossim}(f_i, f_j) = \frac{f_i^T f_j}{|f_i||f_j|}, \quad \mathbf{Pr}(o_i|d_i) = \frac{\exp\big(\text{cossim}(f_{o_i}, f_{d_i})/H\big)}{\sum \exp\big(\text{cossim}(f_{o_j}, f_{d_i})\big)/H}, \tag{2}$$

where H is the temperature to adjust the sensitivity of softmax. The calculation of $\mathbf{Pr}(d_i|o_i)$ follows the similar process. By optimizing Equ. (1), CLIP learns a joint latent space where images and texts are well-aligned relatively. For convenience, we use **CLIP score** to stand for the cosine similarity between text and image features from CLIP.

4 OhMG: Open-vocabulary Human Motion Generation

As shown in Fig. 1, our method includes two stages, i.e., text2pose and pose2motion. For the first stage, we probe the versatile CLIP to distill its text-pose knowledge. In this stage, we found that it's difficult to adjust the pose with the original CLIP via gradient descent. And we propose a novel pipeline distillation to address the problem. For pose2motion, we draw the inspiration from advance nature language model pretraining (Devlin et al., 2018). We pretrain a transformer-based (Petrovich et al., 2021; Vaswani et al., 2017) motion model by mask-and-reconstruction self-supervised learning. And for generating motion in a controllable and flexible manner, we reformulate the condition poses from text2pose stage as prompt (Brown et al., 2020) to the pretrained motion model so as to generate controllable motion given the condition poses. In the following, we elaborate on each part in detail.

4.1 Text2Pose

To generate a pose for text, a well-aligned multi-model feature space is required for the texts and the poses. However, there lacks a multi-model pretrained models for aligning 3D poses and texts. A promising workaround is to render the 3D pose into multi-view images and then use CLIP to measure the alignment between images and text. The illustration of this process is in the upper part of Fig. 2.To address the difficulty of optimization using the complex pipeline as well as reducing the

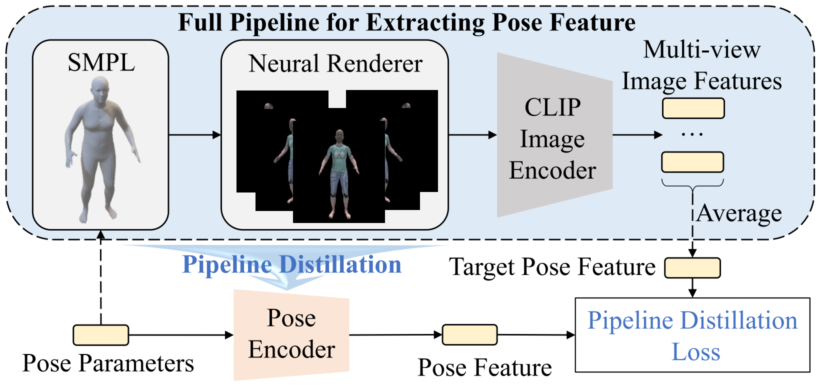

Figure 2: Pipeline distillation. The upper part is the original pipeline for extracting the pose feature for alignment. Our pipeline distillation adopts an end-to-end neural network, i.e., a pose encoder, which takes in the same input and learns to predict the output of the pipeline.

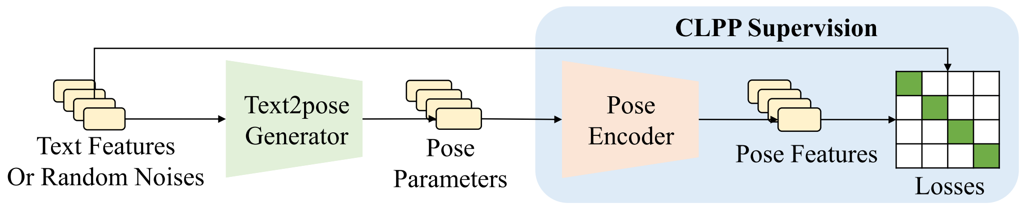

Figure 3: Training process of text2pose generator. During training, either text features extracted from real-world texts or random noises can be used as input features. The text2pose generator takes in the input features and predicts the pose parameters supervised by CLPP.

computation costs, we propose a novel pipeline distillation strategy to learn an efficient and effective pretrained model for 3D pose and text. In the following, we describe how pipeline distillation is conducted. After that, we elaborate on how to use the pretrained model for 3D pose and text to learn a text2pose generator.

Pipeline Distillation. As discussed above, it's non-trivial to conduct gradient descent for optimizing the pose with the original pipeline. In this part, we propose to simplify the complicated pipeline into a concise end-to-end neural network. In other words, we propose a pretrained model for aligning 3D poses and text based on CLIP. To our best knowledge, this work is the first to conduct a distillation strategy to address the optimization problem, and it is also the first work to pretrain a model for aligning 3D pose and text. To distinguish from CLIP, we name our distilled model for 3D pose and text as **CLPP** which stands for **C**ontrastive **L**anguage-**P**ose **P**retraining. Note that, CLPP still uses the original text encoder E_d of CLIP but replacing the image encoder E_o with pose encoder E_p.

To train CLPP, we sample poses from the AMASS dataset (Mahmood et al., 2019), and the poses, p, are processed through the original complex pipeline to obtain the final pose features, denoted as f_p^*. After that, the poses and their pose features are taken as the inputs and targets for training the pose encoder, i.e., E_p. The pipeline distillation loss is formulated as :

$$\mathcal{L}_{E_p}(p, f_p^*) = ||E_p(p) - f_p^*||_2 - \text{cossim}\big(E_p(p), f_p^*\big), \tag{3}$$

where the first term of Equ.(3) is for reducing the element-level distance between the predicted feature and the target feature. While the second term of Equ.(3) is to reduce the angular difference between the features. The overall training sketch is shown in Fig. 2.

Generalized Text2Pose Generator. Unlike the prior work which conducts optimization/matching for different textual descriptions during the inference phase (Hong et al., 2022), our text2pose generator is optimized to handle various text requests and does not require any optimization/matching during inference. Moreover, benefit from low computation cost with CLPP, we manage to optimize the generator using the same loss as CLIP loss (Equ.(1)) which requires a large batch size (Radford et al., 2021b) to construct contrastive losses. Specifically, we replace image with 3D pose and image encoder with pose encoder in CLIP loss (Equ.(1)), resulting in a loss function $\mathcal{L}_{\text{CLPP}}$. By

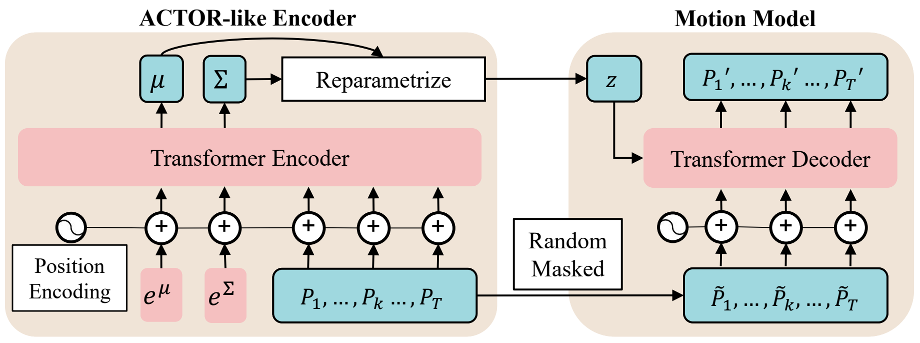

Figure 4: Training process of motion model. On the left is an ACTOR-like Encoder that takes in the original motion and extracts the latent code. The right part is a conditional Motion VAE decoder which takes in randomly masked input and the latent code to reconstruct the original motion.

minimizing the contrastive loss, we found that the optimization process of the text2pose generator is more stable than optimization by maximizing the CLIP score between text and pose. Formally, $\mathbf{f_d} = [f_{d_1}, ..., f_{d_B}]$ denotes a batch of text features extracted using E_d, and G_{t2p} represents the text2pose generator. The text2pose generator G_{t2p} takes text features to predict VPoser latent representation which can be decoded into the angular presentation and fed to E_p to obtain the pose feature. Formally, the loss function is:

$$\mathcal{L}_{\text{t2p}}(G_{\text{t2p}}(\mathbf{f_d}), \mathbf{f_d}) = \mathcal{L}_{\text{CLPP}}\Big(E_p\big(G_{\text{t2p}}(\mathbf{f_d})\big), \mathbf{f_d}\Big) + \lambda_{\text{L2}}||G_{\text{t2p}}(\mathbf{f_d})||, \tag{4}$$

where the second term of the loss function is used to regulate the predicted latent pose to close to the prior distribution of VPoser. From Equ.(4), the only training data are various $\mathbf{d}$ to extract $\mathbf{f_d}$. Fortunately, we can easily collect substantial motion descriptions on the internet. Nevertheless, the collected motion descriptions are inevitably biased by the collection process and therefore hampering the generality of the data. In this paper, we found that it's not necessary to collect real text to extract $\mathbf{f_d}$ for training. Instead, we use random noise features $\mathbf{f_d}$ sampled from a random distribution, e.g. uniform distribution or Gaussian distribution. In our experiment, we found that by training with random sampled $\mathbf{f_d}$, the text2pose generator can be generalized to the real text and obtain surprisingly good performance. This observation reveals the potential of our method to distill other kinds of knowledge from CLIP without collecting textual descriptions.

4.2 Pose2Motion

In our paper, we consider the motion similar to a language sentence and learn a pretrained motion model and reformulate the condition pose as a prompt of the motion model to generate motion in a controllable and flexible manner.

Motion Model Pretraining. For pretraining a motion model, we draw the advanced self-supervised learning of language model (Devlin et al., 2018), which randomly masks a certain proportion of input data and learns to reconstruct the masked data. Our motion model also follows a similar training strategy. Specifically, during training, random proportion of poses of a motion $[p_t]_{t=1:T}$ are masked by a learnable embedding $e^{\text{mask}} \in \mathcal{R}^{|p|}$. Formally, the new input $\tilde{p}_t$ is generated by $\tilde{p}_t = c_t \times p_t + (1 - c_t) \times e^{\text{mask}}$, where $c_t \in \{0, 1\}$ is a binary random condition sampled for each timestep $t \in [1, T]$. When c_t equals to 1, the original pose is preserved. Otherwise, the pose is replaced by the mask embedding.

Then a transformer-based motion model is learn to take in $[\tilde{p}_t]_{t=1:T}$ and predict $[p'_t]_{t=1:T}$ to reconstruct the original motion $[p_t]_{t=1:T}$. However, since the proportion of masked poses is random, it causes the learning process unstable. To this end, we further draw the lessons from the ACTOR (Petrovich et al., 2021) and formulate our pretrained motion model as a conditional VAE. As illustrated in Fig. 4.2, our model includes an ACTOR-like motion encoder and a motion model, where the motion encoder takes in the original motion sequence and predicts the latent for the motion decoder to predict the reconstructed motion. Formally, the loss function is

$$\mathcal{L}_{mm}(p'_t, p_t) = ||p_t - p'_t||_2 + ||\mathcal{M}_{\text{SMPL}}(p_t) - \mathcal{M}_{\text{SMPL}}(p'_t)|| + \lambda_{\text{KL}} KL, \tag{5}$$

where KL is the KL-divergence regularization term to pull the predicted latent to the prior normal distribution. After pretraining the motion model, we only use the motion decoder for pose2motion generation. In the following, we reformulate the poses to be similar to $[\tilde{p}_t]_{t=1:T}$ to prompt the motion model to generate motion.

Motion Prompt. Prompt is a newly rising topic (Brown et al., 2020; Sun et al., 2022; Han et al., 2021) for adapting large foundation model to downstream applications (Jia et al., 2022; Chen et al., 2022). The major motivation of prompt is to reformulate the downstream tasks into the form of the training input of the foundation model. By this means, the pretrained foundation model can be directly used for downstream tasks without finetuning. Inspired by this, with the pretrained motion model, we can reformulate the condition posed by different prompt designs to generate motion in a controllable and flexible manner. Formally, a prompt is a sequence of poses $[\tilde{p}_1]_{t=1:T}$ filled with e^{mask}. And we can change the prompt by replacing a pose $\tilde{p}_k$ of the prompt by the condition pose p, i.e. $\tilde{p}_k = p$. As shown in Fig. 1, we can control the generator to synthesize motion containing the given poses at different positions by designing different prompts.

5 Experiments

We first introduce the datasets and baseline methods used in our experiments. Next, we evaluate the pose generation, including the CLPP and text2pose generator. After that, we investigate the performance of motion generation. The visual results are placed in Appendix. And we strongly suggest the readers view the Appendix for better understanding.

General Settings. We train our model on the AMASS motion dataset (Mahmood et al., 2019). It unifies various optical marker-based mocap datasets by representing them within a common framework. The dataset contains more than 40 hours of motion data without textual labels. Following the same settings in (Hong et al., 2022; Petrovich et al., 2021), we down-sample the motion capture framerate to 30 per second and limit the duration of a motion to 2 seconds. The real-world textual descriptions used in our experiments are obtained from the BABEL motion description dataset (Punnakkal et al., 2021). We remove lengthy descriptions which exceed the CLIP's maximum text length of 77, resulting in a dataset with a size of 4178. For the SMPL model, we input the pose body with global rotation fixed, following the same setting as in AvatarCLIP (Hong et al., 2022). Particularly, the checkpoint of CLIP, i.e. "CLIP-ViT-B/32", is used for extracting both image features and text features. More experimental details such as the network structures and training details could be found in Appendix.

Baselines. In the following, we enumerate the related baseline methods for pose and motion generation, respectively. As for the baseline with the same setting, to our best knowledge, there is only one prior work, i.e., AvatarCLIP (Hong et al., 2022). Therefore, we mostly follow the baselines in AvatarCLIP. To reduce confusion, we use *italic* font to indicate the names of the baselines. To evaluate the performance of text2pose generation, the related baselines are listed in the following. 1) ***Matching*** (Hong et al., 2022) uses CLIP to match among a set of poses according to the texts. The poses are 4096 cluster poses from AMASS using KMeans. 2) ***Optimize*** (Hong et al., 2022) optimizes the pose parameters using the complex pipeline described in our paper to maximize the CLIP score. 3) ***VPoserOptimize*** (Hong et al., 2022) is similar to *Optimize* but optimizing the pose latent code of VPoser instead. As for text/pose to motion generation. There are several approaches: 1) ***Interpolation*** (Hong et al., 2022) linearly interpolates each pair of pose latent codes. 2) ***AvatarCLIP*** (Hong et al., 2022) directly optimizes in the latent space of a decoder of pretrained motion VAE by pulling the generated motion close to the reference poses. 3) ***MotionCLIP*** (Tevet et al., 2022) is training with paired text-motion data. It trains a motion VAE and pushes the latent space close to both of text and image feature spaces of CLIP. And during inference, *MotionCLIP* uses the text features from CLIP as the latent code and decodes it into a motion.

5.1 Text2pose Generation

At the stage of text2pose, there are two modules of interest, i.e., the pipeline distilled model for aligning 3D poses and texts, namely CLPP, and the text2pose generator.

Table 1: The features difference and inference efficiency of CLPP in comparison to the original pipeline. The arrow ↑ indicates the performance is better if the value is larger. Vice versus.

	Batch size ↑	Inference speed (sec) ↓	MSE ↓	Cosine Sim. ↑
Original Pipeline	15	1.2068	-	-
Pipeline Distillation	292350	0.0172	4.56e-4	0.9983

Table 2: Comparison among text2pose baselines. The arrow ↑ indicates the performance is better if the value is larger. Vice versus.

	CLIP Score ↑	In-distrib. ↓	Diverse ↓	Top1 ↑	Top10 ↑	Top50 ↑
Matching	0.2615	0.0150	0.2877	0.0119	0.0821	0.2793
Optimize (I)	0.2446	8.5731	0.0416	0.000	0.0041	0.0138
Optimize	0.2468	8.6403	0.0446	0.0000	0.0041	0.0146
VPoserOptimize (I)	0.2441	0.0133	0.0434	0.0000	0.0025	0.0138
VPoserOptimize	0.2436	0.0142	0.0491	0.0000	0.0019	0.0113
Ours (TS)	0.2580	0.5891	0.0522	0.0117	0.0572	0.1606
Ours (TC)	0.2586	0.6243	0.0446	0.0108	0.0677	0.1620
Ours (NC)	**0.2698**	0.1217	**0.0394**	0.0883	0.3396	0.6261
Ours (NTC)	0.2697	0.1261	0.0402	**0.0993**	**0.3562**	0.6130
Ours (NTLC)	0.2689	**0.0138**	0.0455	0.0828	0.3454	**0.6857**

Effectiveness and Efficiency of CLPP. To measure the difference between CLPP and the original pipeline, we calculate the mean square error (MSE) and the cosine similarity (Cosine Sim.) between predicted features from CLPP and the actual features from the original pipeline. To show the efficiency of CLPP, we measure the maximum batch sizes and the time costs per inference of CLPP and the original pipeline. The results are reported in Tab. 1. From the results, we can observe that the CLPP can extract similar features as the original pipeline with a small mean square error and high cosine similarity between the predicted and the ground-truth features. Nevertheless, CLPP can obtain significant improvement in space occupancy and inference speed by about 20,000x and 700x relative improvement on one NVIDIA V100 Tensor Core (32G).

Evaluation Metrics for Text2Pose Generation. Nevertheless, the ultimate goal of CLPP is to facilitate the training of the text2pose generator for optimizing the average **CLIP score** between the generated poses and the textual descriptions. Besides the CLIP score, we also measure whether the generated poses are within the distribution of the real poses (**In-distrib.**). To calculate In-distrib., we adopt a widely-used pose VAE, i.e., VPoser (Pavlakos et al., 2019), and use the reconstruction errors of generated poses to indicate whether the generated poses are familiar to VPoser. Since VPoser is trained with diverse and substantial real-world poses, the smaller reconstruction errors of the generated poses imply the closer to real-world poses. However, it could happen that the text2pose generator collapses and fail to generate diverse poses according to the textual descriptions. To detect this problem, we also measure the **diversity** of the generated poses by testifying how much information of the textual input is preserved in the generated poses. Specifically, we train a neural network for each text2pose generator by using their generated pose as inputs and the text features as the regression targets. The smaller regression error indicates that more information about the texts is preserved in the generated poses. However, since the CLIP score only considers the relation between the text description and its generated pose, it could happen that the text description might have larger CLIP scores with some other generated poses of other texts. To testify whether the generated poses are distinctive and accurate in a relative manner, we also evaluate the matching accuracy among the text descriptions, which is also known as **CLIP-R-precision** (Park et al., 2021). Specifically, given a textual description, we say the matching is accurate if its generated pose has the largest CLIP score against other generated poses of other texts. Since the texts in BABAL contains many similar textual descriptions, **Top1** cannot comprehensively reflect the actual performance of each generation method. To this end, we also include **Top10** and **Top50** accuracies.

Text2Pose Generation Performances. The evaluation results are represented in Tab. 2. To understand whether optimization makes a difference, we also include the performances of the methods with randomly initialized parameters without optimization, denoted with post-fix (I). We observe that the performance of the optimization-based methods *Optimize* and *VPoserOptimize* remains about same as their initialization, i.e. *Optimize* (I) and *VPoserOptimize* (I). This observation implies that the optimization with the original pipeline is prone to be stuck at the local optimal. As we can see, *Matching* method can obtain a high CLIP score, which implies that the best CLIP score should be at least equal to or higher than 0.2615. And as pointed in Hong et al. (2022), we also observe that by including the VPoser decoder, *VPoserOptimize* (I) can have a better pose initialization and outperforms *Optimize* (I) on the In-distrib. significantly.

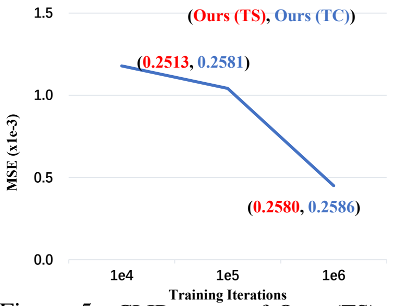

Figure 5: CLIP scores of Ours (TS) and Ours (TC) with CLPP at different iterations.

However, with CLPP, we observed that the optimization manages to improve the CLIP score as well as other metrics. For briefness, among our methods, (T) stands for using real-world texts as training data, (N) stands for using noise features as training data, (S) represents optimization by maximizing CLIP score, (C) represents optimization by minimizing $\mathcal{L}_{\text{CLPP}}$ and (L) means training with L2-norm regularization of Equ.(4). As we can see in Tab. 2, by simply replacing the original pipeline with CLPP, Ours(TS) can significantly improves the CLIP score in comparison with *VPoserOptimize* although Ours(TS) optimize to maximize CLPP score. However, we also observe that Ours(TS) is sensitive to the performance of CLPP. And the stability of optimization can be further improved when minimizing the $\mathcal{L}_{\text{CLPP}}$ instead, i.e., Ours(TC). As shown in Fig. 5, with different training iterations of CLPP, Ours(TC) have stable performance while Ours(TS) varies severely. We contribute such stability to more dense supervision by drawing other samples into the contrastive loss. To this end, we suggest using CLIP-like loss for future experiments. Another interesting and exciting observation is that, by optimizing with noise features instead of text features extracted from real-world texts, Ours(NC) not only outperforms Ours(TC) in CLIP score but also obtains better in-distrib. results. The potential reason for the improvement could be due to the significant increase in the amount and diversity of training data. With a limited amount of training data, it's easier for the generator to exploit the difference between CLPP and the original pipeline to obtain an overfitted solution (e.g., generating strange or twisted poses) for the training data. And combining the real-world text features with noise features, Ours(NTC) ends up with similar performance across different metrics, which implies that only noise features can already generalize to real-world texts. To generate between in-distribution poses, Ours(NTLC) also includes the L2-norm regularization term which pulls the predicted latent pose to the center of the prior distribution of VPoser.

5.2 Motion Generation

As for motion generation, our method pretrains a motion model following language modeling (Devlin et al., 2018) and uses text-consistent poses to prompt the model to synthesize complete text-conditioned motions. Therefore, in this part, we are interested in evaluating the motion generator in three aspects: 1) whether the generated motion contains the condition poses? 2) whether the generated motion follows real-world motion dynamics? 3) whether the generated motion matches the textual description?

Precise Control. To answer question 1, we use the 4096 clustered poses used in AvatarCLIP (Hong et al., 2022) 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 $K-\text{P}$ for a generated motion m conditioned on poses $\{p'_k\}_{j=1:K}$ is formulated as:

$$K-\text{P}(m, \{p'_k\}_{j=1:K}) = Z \times \frac{1}{K} \sum_{k=1:K} \min_{p_j \in m} ||p'_k - p_j||_2, \tag{6}$$

where Z is a constant used for amplifying the result for better reading. Equ.(6) approach zero if the condition poses are more likely to present in the generated motion. As shown on the left of Tab. 3, we observe that *Interpolation* has the best results, which is reasonable since it directly takes the condition poses as a part of the generated motion. The reason why there is still a small error for *Interpolation* is that interpolation is conducted on the latent codes of these poses in the

Table 3: Evaluation for conditional motion generation. The arrow ↑ indicates the performance is better if the value is higher. Vice versus.

	1p↓	2p↓	3p↓	In-distrib.↓	Top1↑	Top10↑	Top50↑
MotionCLIP	-	-	-	0.7943	0.0050	0.0419	0.1649
Interpolation	**0.0900**	**0.0865**	**0.0868**	0.5563	0.0029	0.0184	0.0804
AvatarCLIP	1.9022	2.4353	2.6163	0.3147	0.0014	0.0096	0.0421
Ours	<u>0.6252</u>	<u>0.5219</u>	<u>0.4797</u>	**0.0877**	**0.0689**	**0.2054**	**0.4349**

latent space of VPoser and the encoding-decoding process causes the small error. Between the parameterized methods, i.e., *AvatarCLIP* and ours, our generator achieves significant improvement upon AvatarCLIP. Moreover, our method does not optimize using the condition poses as AvatarCLIP does. Although we cannot outperform *Interpolation*, in practice, we find that it's indistinguishable to visually tell the difference w.r.t. the controllability between *Interpolation* and ours.

Real-world Motion Dynamics. To testify whether the generated motion follows the real-world motion dynamics, we bring a similar measurement from the text2pose, i.e., **In-distrib.**. To this end, we trained an unconditional motion VAE on AMASS. We use BABAL as textual descriptions to generate motions so that non-pose-conditioned *MotionCLIP* can also be compared. As reported in the middle of Tab. 3, our method outperforms the others by a clear margin. Particularly, we found that although *Interpolation* is endowed with the best controllability, its generated motion is constructed through linear interpolation without considering the motion dynamics, resulting in poor In-distrib.. We also notice that the parameterized MotionCLIP performs poorly. The potential reason might be the gap of latent space between MotionCLIP and CLIP text encoder. As shown in Appendix, we observe that MotionCLIP is likely to generate twisted poses while AvatarCLIP and ours are more natural.

Generating Motion According to Texts. Finally, we testify whether the generated motion can reflect the text. Since all of the baseline methods are CLIP-based (except for *Interpolation*), we use CLIP to measure the performance. Specifically, we propose a motion-level CLIP-R-precision. The motion-level CLIP-R-precision calculates the alignment between the textual description and the poses of all motions. We say the matching is accurate if the matched poses in 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, 2) and does not contain irrelevant poses that might cause mismatching for other poses**. From the results at the right of Tab. 3, we find that among all baselines, our method obtains the best TopK accuracies by a clear margin. Among the baselines, we find that *MotionCLIP* performs better, which is reasonable since it trains with pair text-motion data. Even so, ours can still outperform *MotionCLIP* by mining text-pose knowledge from CLIP and using a better motion generation procedure. We also observe that *Interpolation* performs better than *AvatarCLIP*. One of the reasons is that *Interpolation* can better preserve the condition poses than *AvatarCLIP* as we mentioned above. And *Interpolation* is less likely to generate new poses that might distract the matching process.

6 Conclusion

In this paper, we propose a zero-shot open-vocabulary human motion generation (OhMG) framework, which leverages the text-pose knowledge from CLIP to build a text2pose generator and use the generated pose to prompt a pretrained motion model to generate motion. To address the difficulty of optimization with the complex pipeline, we propose a pipeline distillation strategy that manages to distill the text-pose knowledge from CLIP and facilitate the optimization of the text2pose generator. By drawing inspiration from the field of NLP, we also propose a controllable and flexible motion generation framework. Extensive experiments show that our method can generate better text-consistent poses and motions across various baselines and metrics.

REFERENCES

Gunjan Aggarwal and Devi Parikh. Dance2music: Automatic dance-driven music generation. *arXiv: Sound*, 2021.

Hyemin Ahn, Timothy Ha, Yunho Choi, Hwiyeon Yoo, and Songhwai Oh. Text2action: Generative adversarial synthesis from language to action. *international conference on robotics and automation*, 2018.

Chaitanya Ahuja and Louis-Philippe Morency. Language2pose: Natural language grounded pose forecasting. *international conference on 3d vision*, 2019.

Nikos Athanasiou, Mathis Petrovich, Michael J. Black, and Güll Varol. TEACH: Temporal Action Compositions for 3D Humans. In *International Conference on 3D Vision (3DV)*, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Zhongang Cai, Mingyuan Zhang, Jiawei Ren, Chen Wei, Daxuan Ren, Jiatong Li, Zhengyu Lin, Haiyu Zhao, Shuai Yi, Lei Yang, et al. Playing for 3d human recovery. *arXiv preprint arXiv:2110.07588*, 2021.

Zhongang Cai, Daxuan Ren, Ailing Zeng, Zhengyu Lin, Tao Yu, Wenjia Wang, Xiangyu Fan, Yang Gao, Yifan Yu, Liang Pan, et al. Humman: Multi-modal 4d human dataset for versatile sensing and modeling. *arXiv preprint arXiv:2204.13686*, 2022.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web Conference 2022*, pp. 2778–2788, 2022.

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.

Kevin Frans, L. B. Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *arXiv: Computer Vision and Pattern Recognition*, 2021.

Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *arXiv: Computer Vision and Pattern Recognition*, 2021.

Chuan Guo, Xinxin Zuo, Sen Wang, Shihao Zou, Qingyao Sun, Annan Deng, Minglun Gong, and Li Cheng. Action2motion: Conditioned generation of 3d human motions. *acm multimedia*, 2020.

Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5152–5161, June 2022.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

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.

Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. *arXiv*, December 2021.

Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. 2022.

Nikolay Jetchev. Clipmatrix: Text-controlled creation of 3d textured meshes. 2022.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pp. 4904–4916. PMLR, 2021.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. *arXiv preprint arXiv:2203.12119*, 2022.

Xiao Lin and Mohamed R Amer. Human motion modeling using dvgans. *arXiv preprint arXiv:1804.10652*, 2018.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: a skinned multi-person linear model. *international conference on computer graphics and interactive techniques*, 2015.

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.

Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. *international conference on 3d vision*, 2016.

Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. 2022.

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

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.

Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. *arXiv: Computer Vision and Pattern Recognition*, 2021.

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.

Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018.

Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Animatable neural radiance fields for human body modeling. 2021a.

Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9054–9063, 2021b.

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

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

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

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.

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*, 2021a.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021b.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bj
”orn Ommer. High-resolution image synthesis with latent diffusion models. 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar, Seyed Ghasemipour, Burcu Karagol, S Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. 2022.

Aditya Sanghi, Hang Chu, Joseph G. Lambourne, Ye Wang, Chin-Yi Cheng, and Marco Fumero. Clip-forge: Towards zero-shot text-to-shape generation. *arXiv: Computer Vision and Pattern Recognition*, 2021.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pp. 894–906. PMLR, 2022.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

Tian-Xiang Sun, Xiang-Yang Liu, Xi-Peng Qiu, and Xuan-Jing Huang. Paradigm shift in natural language processing. *Machine Intelligence Research*, 19(3):169–183, 2022.

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.

Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. *computer vision and pattern recognition*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual description. *arXiv preprint arXiv:2210.02399*, 2022.

Yael Vinker, Ehsan Pajouheshgar, Jessica Y. Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. Clipasso: Semantically-aware object sketching. 2022.

Timo von Marcard, Roberto Henschel, Michael J. Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate {3D} human pose in the wild using {IMUs} and a moving camera. *european conference on computer vision*, 2018.

Kim Youwang, Kim Ji-Yeon, and Tae-Hyun Oh. Clip-actor: Text-driven recommendation and stylization for animating human meshes. *arXiv preprint arXiv:2206.04382*, 2022.

Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *arXiv preprint arXiv:2208.15001*, 2022.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753, 2019.

Yufan Zhou, Ruiyi Zhang, Changyou Chen, Chunyuan Li, Chris Tensmeyer, Tong Yu, Jiuxiang Gu, Jinhui Xu, and Tong Sun. Lafite: Towards language-free training for text-to-image generation. 2022.

A Model Structure and Training Details

Our model consists of two generators, i.e., text2pose and pose2motion generator. To optimize the text2pose generator, we also distill a text-pose model, namely CLPP, from the versatile CLIP. Therefore, there are mainly three neural networks in this paper, i.e., CLPP, text2pose, and pose2motion generator. 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, and 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 the similar architecture as Bert Devlin et al. (2018). The architectures of the transformer encoder and decoder layer are implemented by PyTorch Paszke et al. (2019). The poses of a motion are first projected by a linear projection layer, then processed by a is 8-layered transformer encoder/decoder, and finally fed to an estimation layer to obtain to prediction. The number of attention heads is 8, the hidden size is 1024 and the dropout rate is 0.1.

CLPP. CLPP is distilled from CLIP for aligning 3D poses and texts. Specifically, for the text encoder, CLPP simply reuses the text encoder of CLIP. As for the pose encoder, CLPP adopts the ResNet-based network. CLPP 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-5 at the beginning and annealed by the CosineAnnealingLR scheduler implemented by PyTorch. The number of training epochs is 1M. As for the process of the **original pipeline**, we mostly adopt the process used in AvatarCLIP Hong et al. (2022). Specifically, as shown in Fig. 2, the 3D pose representation is first used to generate the 3D meshes by SMPL. 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.

Text2Pose Generator. The architecture of our text2pose generator is the ResNet-based network. It takes the text features extracted by CLPP/CLIP's text encoder and predicts the latent pose of 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 CLPP pose encoder for the pose feature. The batch size is 1024, 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 iterations of each epoch is 1K. λ_{L2} is set as 0.15 empirically. 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. If the generator is trained with both noise features and text features, the proportions of the noise features and text features are 50% and 50%.

Pose2Motion Generator. As described in the main text, our pose2motion generation uses the combination of a pretrained motion model and pose prompt. And the pretrained motion model is the only network in this stage. The pretrained motion model uses the transformer-based network architecture. As shown in Fig. 4.2, the pretrained motion model includes an ACTOR-like encoder and a motion model. During the training phase, the ACTOR-like 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 model 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 10K. λ_{KL} is set as 1e-4 empirically. During inference, the values of the latent code are set as 0.

B Experiment Details

B.1 Baselines Details.

The results of all baseline methods are obtained by running their open-released codes.

B.2 Evaluation Metrics.

In our experiments, all results are obtained in an NVIDIA V100 Tensor Core (32G). As mentioned, the results are multiple by a constant Z to remove zeros for better reading. Specifically, Z for **In-distrib.** and **Diverse** in Tab. 2 is 10. Z for $K-\text{P}$ and **In-distrib.** in Tab. 3 is 100.

B.3 Discussion and Limitations

As foundation models 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. The reason why we investigate leveraging the foundation model to zero-shot motion generation is that we consider the foundation model as a promising world model of the real world, and the motion generation is one of the preliminary tasks for the agent to learn to interact with the real world.

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. Using model dynamics learned from motion data can only compensate to a limited extent. It cannot handle some difficult texts like a sentence having multiple successive motions. Although this can be addressed by a divide-and-conquer strategy, it is a band-aid solution and resolves the problems once and for all. We suppose the best practice is to use a better foundation model that can handle the temporal description. 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.

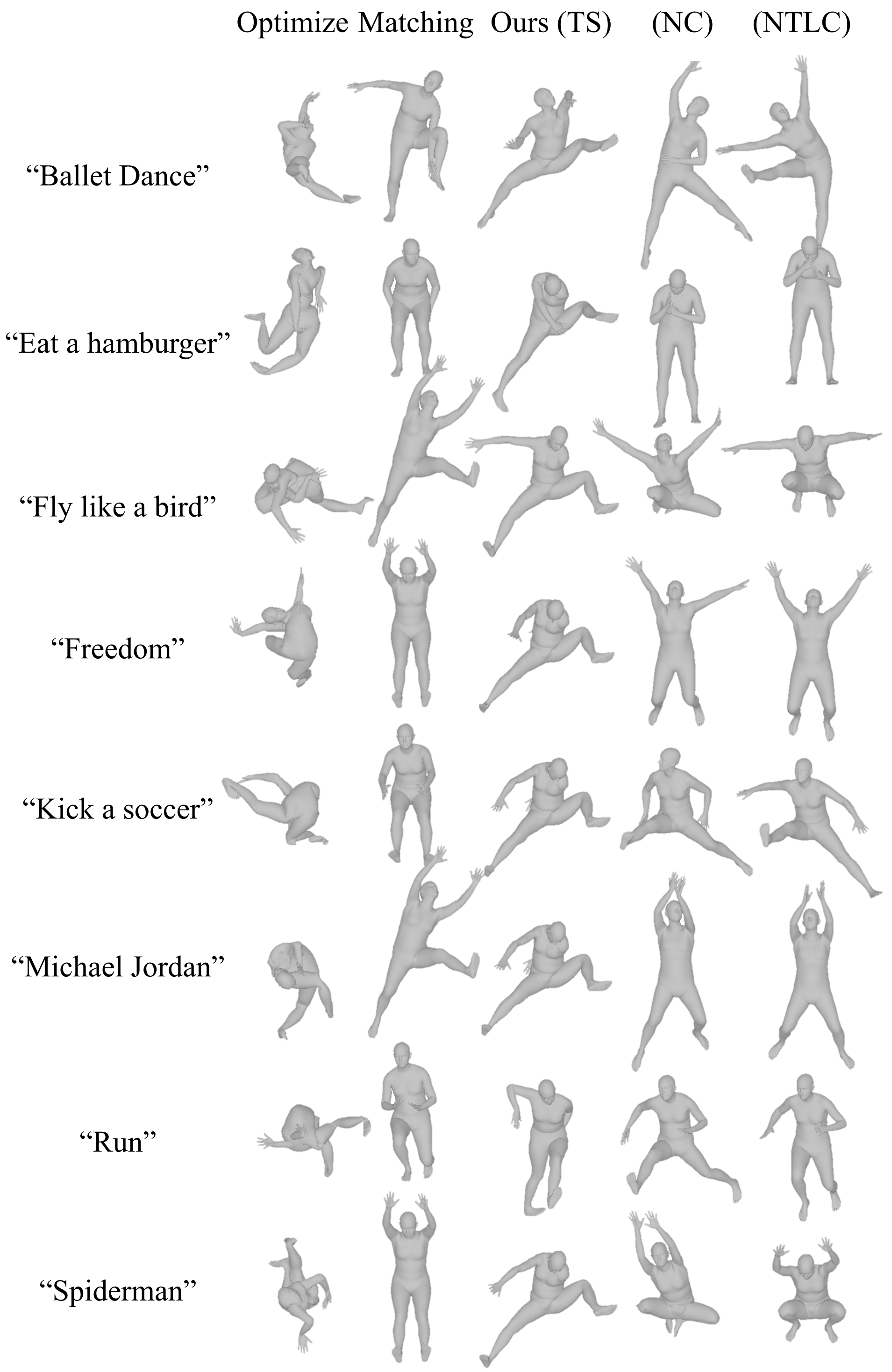

Figure 6: Visual results of text2pose baselines.

B.4 Visual Results

In this part, we show the visual results of each baseline. The comparison of text2pose is shown in Fig. B.4 and the comparison of text2motion generation is shown in Fig. B.4.

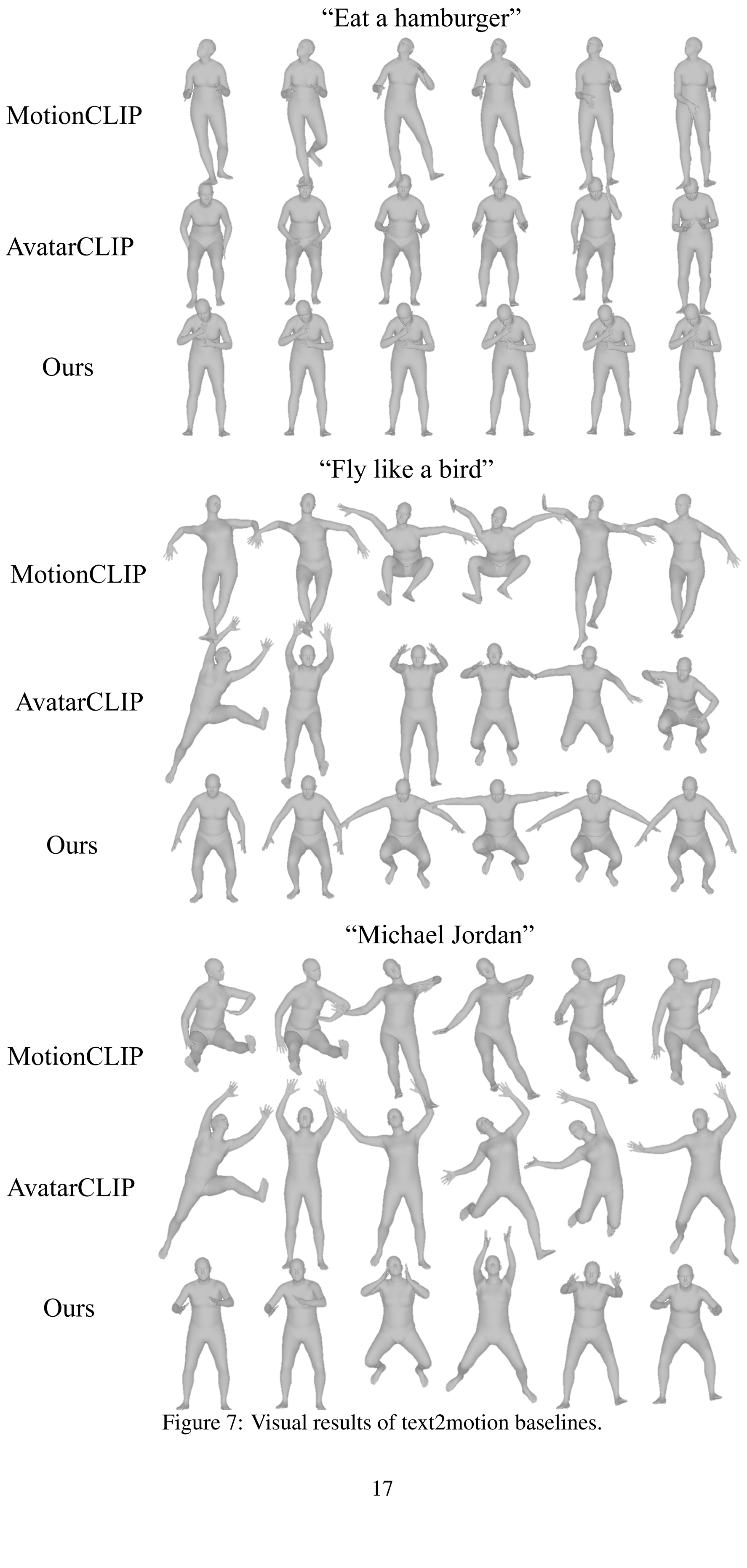

Figure 7: Visual results of text2motion baselines.