Zero-Shot Text-to-Parameter Translation for Game Character Auto-Creation

Rui Zhao¹, Wei Li², Zhipeng Hu¹, Lincheng Li¹, Zhengxia Zou³, Zhenwei Shi³, Changjie Fan¹
¹Netease Fuxi AI Lab, ²Nankai University, ³Beihang University
{zhaorui10, zphu, lilincheng, fanchangjie}@corp.netease.com,
liwei@dbis.nankai.edu.cn, {zhengxiazou, zhenweishi}@buaa.edu.cn

Figure 1. Game characters created by the proposed text-to-parameter translation (T2P) given different text prompts. The front view and three side views are shown for each character.

Abstract

Recent popular Role-Playing Games (RPGs) saw the great success of character auto-creation systems. The bone-driven face model controlled by continuous parameters (like the position of bones) and discrete parameters (like the hairstyles) makes it possible for users to personalize and customize in-game characters. Previous in-game character auto-creation systems are mostly image-driven, where facial parameters are optimized so that the rendered character looks similar to the reference face photo. This paper proposes a novel text-to-parameter translation method (T2P) to achieve zero-shot text-driven game character auto-creation. With our method, users can create a vivid in-game character with arbitrary text description without using any reference photo or editing hundreds of parameters manually. In our method, taking the power of large-scale pre-trained multi-modal CLIP and neural rendering, T2P searches both continuous facial parameters and discrete facial parameters in a unified framework. Due to the discontinuous parameter representation, previous methods have difficulty in effectively learning discrete facial parameters. T2P, to our best knowledge, is the first method that can handle the optimization of both discrete and continuous parameters. Experimental results show that T2P can generate high-quality and vivid game characters with given text prompts. T2P outperforms other SOTA text-to-3D generation methods on both objective evaluations and subjective evaluations.

1. Introduction

Role-Playing Games (RPGs) are praised by gamers for providing immersive experiences. Some of the recent popular RPGs, like Grand Theft Auto Online¹ and Naraka², have opened up character customization systems to players. In such systems, in-game characters are bone-driven and controlled by continuous parameters, like the position, position, and face expression.
rotation, scale of each bone, and discrete parameters, like the hairstyle, beard styles, make-ups, and other facial elements. By manually adjusting these parameters, players can control the appearance of the characters in the game according to their personal preferences, rather than using predefined character templates. However, it is cumbersome and time-consuming for users to manually adjust hundreds of parameters - usually taking up to hours to create a character that matches their expectations.

To automatically create in-game characters, the method named Face-to-parameter translation (F2P) was recently proposed to automatically create game characters based on a single input face image [38]. F2P and its variants [39, 41] have been successfully used in recent RPGs like Narake and Justice, and virtual meeting platform Yaotai. Recent 3D face reconstruction methods [2, 7, 26, 33, 42–44] can also be adapted to create game characters. However, all the above-mentioned methods require reference face photos for auto-creation. Users may take time to search, download and upload suitable photos for their expected game characters. Compared with images, text prompts are more flexible and time-saving for game character auto-creation. A very recent work AvatarCLIP [10] achieved text-driven avatar auto-creation and animation. It optimizes implicit neural networks to generate characters. However, the created characters are controlled by implicit parameters, which lack explicit physical meanings, thus manually adjusting them needs extra designs. This will be inconvenient for players or game developers to further fine-tune the created game characters as they want.

To address the above problems, we propose text-to-parameter translation (T2P) to tackle the in-game character auto-creation task based on arbitrary text prompts. T2P takes the power of large-scale pre-trained CLIP to achieve zero-shot text-driven character creation and utilizes neural rendering to make the rendering of in-game characters differentiable to accelerate the parameters optimization. Previous works like F2Ps give up controlling discrete facial parameters due to the problem of discontinuous parameter gradients. To our best knowledge, the proposed T2P is the first method that can handle both continuous and discrete facial parameters optimization in a unified framework to create vivid in-game characters. F2P is also the first text-driven automatic character creation suitable for game environments.

Our method consists of a pre-training stage and a text-to-parameter translation stage. In the pre-training stage, we first train an imitator to imitate the rendering behavior of the game engine to make the parameter searching pipeline end-to-end differentiable. We also pre-train a translator to translate the CLIP image embeddings of random game characters to their facial parameters. Then at the text-to-parameter translation stage, on one hand, we fine-tune the translator on un-seen CLIP text embeddings to predict continuous parameters given text prompt rather than images, on the other hand, discrete parameters are evolutionally searched. Finally, the game engine takes in the facial parameters and creates the in-game characters which correspond to the text prompt described, as shown in Fig 1. Objective evaluations and subjective evaluations both indicate our method outperforms other SOTA zero-shot text-to-3D methods.

Our contributions are summarized as follows:

1) We propose a novel text-to-parameter translation method for zero-shot in-game character auto-creation. To the best of our knowledge, we are the first to study text-driven character creation ready for game environments.

2) The proposed T2P can optimize both continuous and discrete parameters in a unified framework, unlike earlier methods giving up controlling difficult-to-learn discrete parameters.

3) The proposed text-driven auto-creation paradigm is flexible and friendly for users, and the predicted physically meaningful facial parameters enable players or game developers to further fine-tune the game character as they want.

2. Related Work

2.1. Parametric Character Auto-Creation

Character auto-creation has been an emerging research topic because of its significance in role-playing games, augmented reality, and metaverses. Some methods on this topic are recently proposed. Tied Output Synthesis (TOS) learns to predict a set of binary facial parameters to control the graphical engine to generate a character that looks like the human in input photo [49]. Face-to-Parameter translation (F2P) is proposed to optimize a set of continuous facial parameters to minimize the distance between the generated game character’s face and the input photo [38]. In F2P’s following works [39, 41], the framework is improved to achieve fast and robust character creation. The PockerFace-Gan is proposed to decouple the expression features and identity features in order to generate expression-less game characters [40]. Borovikov et al. applies domain engineering and predict the facial parameters in a global-local way, considering the face as a hierarchical ensemble of general facial structure and local facial regions [3]. These methods all need reference photos to create characters, while we aim at creating characters based on text input.

2.2. 3D Face Reconstruction

3D face reconstruction also aims to generate a 3D face given single or multi-view 2D facial images. 3D morphable model (3DMM) [1] and its variants [2,6,9,12,19] are representative methods in the literature. They first parameterize a 3D face mesh data and then optimize it to match the facial identity, expression, and texture of given reference im-
Figure 2. An overview of the proposed T2P. $E_i$ and $E_f$ denote the CLIP image encoder and text encoder, respectively. An imitator is trained to mimic the game engine and achieve differentiable rendering. A translator is pre-trained to translate the CLIP image embeddings to continuous facial parameters. When creating game characters given text prompts, T2P searches continuous facial parameters by fine-tuning the translator and searches discrete facial parameters by the evolution search. Finally, the facial parameters are fed into the game engine to render the in-game characters.

3. Method

Fig. 2 shows an overview of the proposed T2P. We first train an imitator to simulate the game engine and pre-train a translator to translate the CLIP image embeddings to continuous facial parameters. Then, to achieve text-to-parameter translation, given the text prompts, we fine-tune the translator to predict continuous parameters and combine the evolution search to optimize discrete parameters.

3.1. Imitator

We train a neural imitator to mimic the behavior of the game engine in order to differentiate the rendering of in-game characters. It takes in continuous facial parameters $\mathbf{x}$ and renders the front view of the game character $\mathbf{y}$. Different from the F2P [38] taking a similar generator network architecture of DC-GAN [29], we add a positional encoder at the input-end of the renderer to improve the facial param-
We aim to train a translator to predict continuous facial parameters based on CLIP text embeddings. To reduce the learning difficulty, we first pre-train the translator on CLIP image embeddings and then fine-tune it on text CLIP embeddings. The main reason is that text-parameter pairs are expensive to collect, while image-parameter pairs can be obtained in an efficient manner.

To prepare the training data, we randomly sample 170K continuous facial parameters \( \mathbf{x} \) from a multidimensional uniform distribution \( u(\mathbf{x}) \). We feed these parameters into the game engine to render out the facial images. Then these facial parameters and image pairs are split into 80% and 20% for training and validation.

### 3.2. Continuous Parameters Searching

We take the randomly sampled facial parameters \( \mathbf{x} \) and use L1 loss as the loss function to train the imitator:

\[
\mathcal{L}_G(\mathbf{x}) = E_{\mathbf{x} \sim u(\mathbf{x})} \{ || \mathbf{y} - \hat{\mathbf{y}} ||_1 \} = E_{\mathbf{x} \sim u(\mathbf{x})} \{ || G(\mathbf{x}) - \text{Engine}(\mathbf{x}) ||_1 \},
\]

where \( G(\mathbf{x}) \) and \( \text{Engine}(\mathbf{x}) \) represent the image rendered by the imitator and the game engine, respectively.

To prepare the training data, we randomly sample 170K continuous facial parameters \( \mathbf{x} \) from a multidimensional uniform distribution \( u(\mathbf{x}) \). We feed these parameters into the game engine to render out the facial images. Then these facial parameters and image pairs are split into 80% and 20% for training and validation.

### 3.3. Discrete Parameters Searching

In the bone-driven face model, besides continuous facial parameters controlling its bones, discrete facial elements (like the hairstyle, beard styles, and make-up) are also important. However, these elements are difficult for the imitator to learn, because they are discrete and highly changeable. Unlike previous methods that ignore discrete parameters during optimization, we propose to evolutionally search them by directly interacting with the game engine. Evolutionary algorithms have been widely used in reinforcement learning and neural architecture search [21, 36], where the objective function can be optimized without using any gradient information.

When T2P creates game characters given text prompts, there is no image embeddings available. Though the CLIP is trained to pull the text and image pairs close to each other in the embedding space, there are still gaps between the two modalities. We, therefore, fine-tune the translator to fit the input text embeddings. Inspired by the recent prompt tuning study [53], we fix the parameters of the transformer and fine-tune a tiny tuner head. The translator is trained to map the text embeddings \( e_T \) to facial parameters \( \mathbf{x} \). Then the facial parameters are fed into the imitator to render the image of the game character. The fine-tuning objective function is to minimize the cosine distance between the given text embeddings \( e_T \) and the image embeddings of the rendered image:

\[
\mathcal{L}_{\text{CLIP}}(e_T, \mathbf{x}) = 1 - \cos(e_T, E_I(G(\mathbf{x}))) = 1 - \cos(e_T, E_I(G(F(e_T)))),
\]

where \( E_I \) is the CLIP image encoder. The parameters of the fine-tuned head \( w \) are iteratively updated as follows:

\[
w \leftarrow w - \eta_t \frac{\partial \mathcal{L}_{\text{CLIP}}}{\partial w},
\]

where \( \eta_t \) is the learning rate at \( t \)th iteration. We follow the snapshot ensembles [11] and set the learning rate using the cosine annealing schedule with warm restarts (SGDR) [22] to encourage the translator to converge to and escape from local minima:

\[
\eta_t = \eta_{\text{min}} + \frac{1}{2}(\eta_{\text{max}} - \eta_{\text{min}})(1 + \cos(N_t / N \pi)),
\]

where \( \eta_{\text{min}}, \eta_{\text{max}}, \) and \( \eta_t \) denote the minimum, maximum, and current learning rate, respectively. \( N \) denotes the number of iterations between two warm restarts, and \( N_t \) denotes the number of iterations since the last restart. Each time the \( N_t \) equals \( N \), the current iteration is a snapshot point, and we save the predicted facial parameters at this point. These facial parameters are then used to initialize the first population of the evolution search.
dom noise at multiple randomly selected position

are scored by the CLIP model as follows,

Transformer encoder layers [45], each of them having eight

tanh activation of the output layer is removed to encour-

erator is similar to DCGAN’s generator [29], except that its

gerator with six transposed convolution layers. The gener-

tional encoder with four fully-connected layers and a gen-

3.4. Implementation Details

until the CLIP score is converged.

next generation and get involved in the looping selection,

the better ones of the parents’ parameters are selected as the

The newly generated children’s parameters together with

The evolution process terminates

until the CLIP score is converged.

3.4. Implementation Details

Network architecture. Our imitator consists of a posi-

ional encoder with four fully-connected layers and a gen-

erator with six transposed convolution layers. The gener-

ator is similar to DCGAN’s generator [29], except that its

Tanh activation of the output layer is removed to encour-

age a better convergence. The translator consists of eight

Transformer encoder layers [45], each of them having eight

multi-attention heads, and sixteen input tokens. The first to-

token is the CLIP embeddings and the other tokens are learn-

able. We concatenate a prediction head with one single

fully-connected layer after the Transformer. The fine-tuning

head of the translator is a three layers perceptron with a bot-

tleneck architecture.

Training details. The imitator and translator are both

trained using SGD optimizer [4]. We set the momentum to

0.9 and set the weight decay to 5e-4. For imitator pretrain-

ing, the learning rate is set to 1e-3 and is reduced to 0.98x

per 30 epochs, and the training is stopped after 500 epochs.

For translator pre-training, the learning rate is set to 1e-4

and is reduced to 0.1x at the 600th epoch and the training

is stopped at the 1000th epoch. We randomly sample

170K facial parameters and corresponding rendered images

of in-game characters pairs to train the imitator and transla-

tor. For translator fine-tuning, the minimum and maximum

learning rates are set to η_{min} = 0 and η_{max} = 1, respec-
tively, and the number of iterations between two warm starts

N is set to 10 for the SGDR learning rate scheduler. Fine-
tuning is stopped when the CLIP scores are no longer im-

proved by more than 100 iterations.

Evolution search. The facial parameters predicted by

the translator at the last 5 snapshot points are selected as

initial values. Each set of facial parameters contains 269

continuous parameters and 62 discrete parameters, and the

initialized values of these discrete parameters are set to ze-

ros, which means these facial elements do not appear at

the beginning. These 5 sets of facial parameters together

with 5 more random ones are the first population for the

evolution search. We found that updating continuous pa-

rameters together with discrete parameters in the evolution

search achieves better results. The number of selected pairs

of parents is set to 10. The weight coefficient α is set to 0.8.

The crossover rate is set to 0.4 and the mutation rate is set

to 0.05.

Prompt engineering. To enhance the text prompts, we

follow the CLIP [28] and adapt prompt ensembling to the
given text prompts. We preset 12 template sentences, such

as “{} head rendered in a game engine”, and then fill the

Figure 4. Game characters created by the proposed T2P given the text prompt “monkey”. The first five game characters are created by the translater at different fine-tuning iterations. The last one is created by the evolution search, adding a discrete facial element, a beard.
Figure 5. In-game fictional characters created by the proposed T2P given different text prompts. The results in the first row are created by the translator. The results in the second row are created by the evolution search.

“{}” with the input text prompt. We calculate the CLIP text embeddings of the filled sentences and take their mean value as the input text embeddings for the translator and evolution search. For evolution search, we further add “side view of” to the template sentences when calculating the CLIP score of the rendered images of the side view.

4. Experimental Results and Analysis

4.1. Game Character Auto-Creation

Fig. 4 shows the game characters created by T2P given the text prompt “monkey”. The first five images show the in-game characters created by the translator at different fine-tuning iterations. The in-game character gradually grows from a normal human face to look like a monkey. The evolution search further searches discrete facial elements and also slightly improves continuous parameters. The last image of Fig. 4 shows the evolution search adds a beard to the character to make it more vivid. In this process, the proposed T2P is enabled to search both continuous and discrete facial parameters to optimize the in-game character to be consistent with the given text prompt and vivid. Fig. 5 shows more results of fictional character creation. Results in the first row are controlled by continuous parameters, and results in the second row are added discrete facial elements.

T2P can create characters with animal heads, as shown in Fig. 4, fictional characters, as shown in Fig. 5, and celebrities, as shown in Fig. 6, and characters conditioned on compacted text prompts, as shown in Fig. 7. These results show the powerful zero-shot game character auto-creation ability of the proposed T2P. By inputting only a text prompt, T2P can generate a vivid character, which is more flexible and time-saving for players or game developers compared to manual customization.

4.2. Comparison with Other Methods

We compare the proposed method with AvatarCLIP [10] and DreamFusion [27]. The comparison includes objective evaluations and subjective evaluations. Since DreamFusion is not open source yet, we use the community implementation version of it, named Stable-DreamFusion. This version uses the open-source stable diffusion model [34] to drive the 3D object generation. We only compare the heads generated by these methods. This may introduce unfairness, thus we will never claim superiority besides the head part.

We feed 24 different text prompts into these two methods and our proposed T2P to generate characters respectively. Three examples are shown in Fig. 8. For objective evaluations, we compare the Inception Score [35], CLIP Ranking-1, and their speed (run on NVIDIA A30), as shown in Table 1. For each method, CLIP Ranking-1 calculates the ratio of its created characters ranked by CLIP as top-1 among the characters created by all three methods. The evaluation scores show the proposed T2P outperforms the other two methods and runs at a much faster speed.

For subjective evaluations, we invite 20 volunteers to evaluate the generation results in terms of realistic degree and consistency with the given text. They are asked to focus on the heads and faces of the characters and score them from 1 to 5, where 1 is the worst and 5 is the best. The evaluation results are shown in Table 1. Evaluation results show our method consistently outperforms the other two methods. We also notice that AvatarCLIP performs good at celebrities generation, Dreamfusion is good at fictional characters generation, while our method performs better at both types, just as shown in Fig. 8.

4.3. Ablation Studies

We conduct ablation studies to analyze the importance of the proposed translator and evolution search. We run our framework with three settings, including 1) only evolution search 2) only translator and 3) both translator and evolution search. The details of these settings are as follows.

1) Evolution Search. The translator is removed from the framework and the evolution search is used to directly search both continuous and discrete facial parameters given text prompts.

2) Translator. The evolution search is abandoned, and the translator is fine-tuned to translate the given text prompts into continuous facial parameters and gives up controlling discrete parameters.

3) Full Implementation. Given text prompts, the translator is fine-tuned to predict continuous facial parameters. Then, the evolution search further searches discrete parameters and also improves the continuous ones.

Fig. 9 shows the CLIP scores increasing curves with the
Figure 6. In-game celebrities created by the proposed T2P. This figure shows the front view and the side view for each character.

Table 1. Comparison results of DreamFusion, AvatarCLIP, and the proposed T2P in terms of objective and subjective evaluations.

<table>
<thead>
<tr>
<th>Method</th>
<th>Objective Evaluations</th>
<th>Subjective Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inception Score ↑</td>
<td>CLIP Ranking-1 ↑</td>
</tr>
<tr>
<td>DreamFusion</td>
<td>1.60 ± 0.12</td>
<td>16.67%</td>
</tr>
<tr>
<td>AvatarCLIP</td>
<td>1.37 ± 0.31</td>
<td>16.67%</td>
</tr>
<tr>
<td>T2P (ours)</td>
<td>1.65 ± 0.21</td>
<td>66.66%</td>
</tr>
</tbody>
</table>

T2P running in 300 seconds. The means and standard deviations are calculated based on 100 times repeat running driven by one text prompt. As shown in the figure, the full implementation of our method always outperforms the other two. The translator is optimized rapidly to find optimal continuous parameters but can not further improve the CLIP scores because of lacking discrete facial elements. Compared with the translator, the evolution search is quite slow but can reach a higher CLIP score. The full implementation of T2P takes advantage of both translator and evolution search and achieves fast and better optimization.

We further test different settings of proposed T2P on 100 different text prompts to evaluate their performance. Table 2 shows the results. The first row is the result of directly using the pre-trained translator to predict continuous facial parameters, and the second row is the result of fine-tuning translator to predict parameters. The fine-tuned one can achieve a higher CLIP score, which indicates the necessity of fine-tuning. The CLIP scores of only using the evolution search and the full version of T2P are shown in the third and fourth rows, respectively. The full version of T2P achieves the highest CLIP score because it can search both continuous and discrete facial parameters to create better in-game characters.
4.4. Facial Parameter Interpolation

Since the generated characters are controlled by parameters with explicit physical meanings, users can further ad-
just the outlook of the characters as they want. One can also interpolate different facial parameters to create a new character, as shown in Fig. 10. The first row shows the inter-
polation between the monkey and Thanos, in which the new facial parameters are calculated as follows,

\[ x_{\text{new}} = \beta x_{\text{monkey}} + (1 - \beta) x_{\text{Thanos}}, \]

(9)

where \( \beta \) is the interpolation coefficient decreasing from 1 to 0. The results in the second row of Fig. 10 show the inter-
polation between the monkey and Shrek. Besides, more than two characters can also be interpolated. We believe the ben-
efits of the facial parameters controlling bone-driven game characters can give players a higher degree of freedom in character customization.

5. Conclusion

We propose a novel method called “text-to-parameter translation” to create bone-driven in-game characters given text prompts. Our method achieves high-quality zero-shot creation of in-game characters and can search both con-
tinuous and discrete facial parameters in a unified framework. The proposed text-driven framework is flexible and time-saving for users, and the created bone-driven charac-
ters with physically meaningful facial parameters are conve-
nient for users to further edit as they want. Experimental re-
results show our method achieves high-quality and vivid zero-
shot text-driven game character auto-creation and outper-
forms other SOTA text-to-3D generation methods in terms of objective evaluations, speed, and subjective evaluations.
References


