An Interesting Fighting Game using Style-based Generative Adversarial Network and Real-time Human Pose Estimation (Mid.)

Lin-Yung Hsieh
National Tsing Hua University
s111061553@m111.nthu.edu.tw

Jin-Cheng Jhang
National Tsing Hua University
s111061540@m111.nthu.edu.tw

Yu-Cheng Hsieh
National Tsing Hua University
s111061517@m111.nthu.edu.tw

Yi-Shan Lee
National Tsing Hua University
s110061623@m110.nthu.edu.tw

You-Ze Huang
National Tsing Hua University
s111033545s@m111.nthu.edu.tw

1. Introduction

We want to implement a 2-people fighting game. When entering the game, two players will take a selfie respectively as their player avatar in the game. During the game, the camera will capture the players' human pose, reflecting the human pose in the game. When the player raises his/her right hand, for example, the avatar will go right.

Besides, as a player is attacked in the game, the face of avatar (we call it source face) will gradually change into another face (target face). The more health points a player loses, the more his/her avatar will change. The player should try his/her best in order to keep his/her face from turning into other people's face.

2. Technical Part

Figure 1 shows the flow chart of our game. We'll introduce **styleGAN**, **human pose estimation** and **game interface** in this section.

2.1. StyleGAN

In order to mix the style of two real faces, we utilize the style encoding [6], style mixing, latent code interpolation and StyleGAN [4] [5] inference. Firstly, Style encoding aims to find a corresponding latent code w from a real image I such that the generated image I' from w is very similar to the original real image I (you can regard this task as a reconstruction task). We use Style encoding to transform source images I_{source} and target image I_{source} to obtain corresponding latent vectors w_{source} and w_{target} and then use these latent vectors to do latent code interpolation.

Secondly, the main idea of latent code interpolation is to get a sequence of latent codes $[w_1, ..., w_n]$ by interpolating different proportion of latent code w_{source} and w_{target}, where n is the length of sequence (e.g. w_1 consists of 95% of w_{source} and 5% of w_{target} ; w_2 consists of 90% of w_{source} and 10% of w_{target}...)

Finally, we will put results from latent code interpolation $[w_1, ..., w_n]$ into StyleGAN model and the model outputs a sequence of images $[I_1, ..., I_n]$ that mixed the style of w_{source} and w_{target}. From the left part of the sequence $I_1, I_2, ...$ preserves more information of w_{source} while the right part of the sequence $I_n, I_{n-1}, ...$ preserves more information of w_{target}. In our game, we will change the player's face base on image sequences $[I_{source}, I_1, I_2, ..., I_n, I_{target}]$ whenever the player loses health points.

2.2. Human pose estimation

In order to reflect the human action in the game, we need a model that can predict real-time human pose. Besides, due to the immediacy we require in our game and the laptop we use without the high Compute Capability GPU, the model cannot be too computationally expensive. Therefore, we discard some state-of-the-art human pose estimation frameworks such as YOLOv7 [7].

Instead, we select **mediapipe** [2] developed by **Google** to achieve real-time human pose estimation. It captures the real-time image from the camera and performs real-time single person human pose estimation, which can predict 33 key points as Figure 2 shows.

In order to predict two people's key points in one image, we crop the image into two halves. One per-

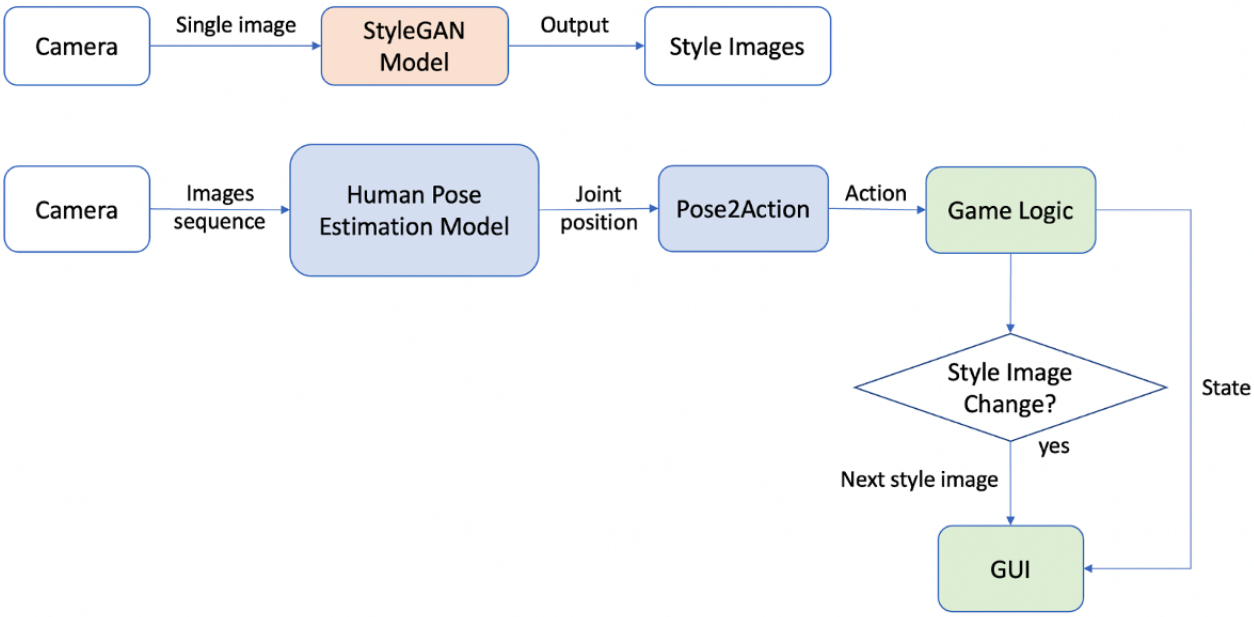

Figure 1. System flow chart. Players will take a selfie respectively as their player avatar before the game starts. During the game, the camera will capture the players' human pose, reflecting the human pose in the game. The faces in the game will also change by time according to the health points of each player.

son stands on the left side and the other stands on the right side. We perform their pose estimation individually. The position matrix of the left-side person is denoted as $M_{left} = [[x_1, y_1], [x_2, y_2], ..., [x_{33}, y_{33}]]$ and the position matrix of the right-side person is denoted as $M_{right} = [[x_1, y_1], [x_2, y_2], ..., [x_{33}, y_{33}]]$, where x and y are the positions of each key point. We can use these 33 key points to predict actions such as "raise right hand." Take "raising right hand" for example. When the position y of key point 15, the right wrist, is larger than the position y of key point 0, the nose, we can define the action as "raising right hand." This is how the pose-to-action step work.

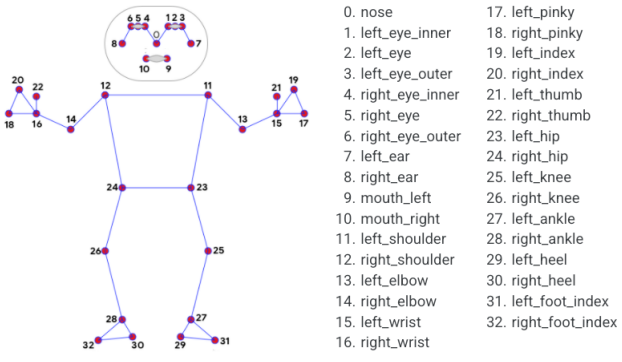

Figure 2. 33 key points predicted by mediapipe.

2.3. Game interface

In this part, we will need to create the GUI of this game. At the beginning of the game, two players will first take a selfie through the camera, and the pictures will become the faces of the characters in the game and show on the screen. Next, when the game starts, we will receive the real-time actions from the previous part of human pose estimation (Sec. 2.2), then the game characters will do the corresponding actions on the screen.

Throughout this game, whenever the character is attacked, his/her health point will drop. Meanwhile, we will switch his/her face to the next image which comes from the sequence of the player's style images (Sec. 2.1). The game will come to the end if one of the character's face completely become the pre-selected target image, in another words, the health point equals to zero.

2.4. Evaluation

We will evaluate our interesting game from some aspects, including **per-pixel reconstruction error**, **inference time of styleGAN**, **accuracy of actions**, and **game performance**.

In order to make the mixed faces much similar to the fusion of source face and target face, we utilize per-pixel reconstruction error to evaluate performances of style encoding. To be more specific, we will use mean absolute error (MAE) as an evaluation metric, where y_j denotes every

pixel in image, and n is total pixels in image.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y_j}| \quad (1)$$

To evaluate the inference time of styleGAN, we'll use following pseudo code to get the inference time directly.

```
start = time.time()
# inference styleGAN
end = time.time()
runtime = end - start
```

As for accuracy of actions, we'll use Table 1 to perform some selected actions (e.g. raising right hand) for 10 times, observing how many times an action is correctly classified by our game and calculate the corresponding accuracy.

Action	# of test case	# of correctly detected case	Accuracy (%)
raise left hand	10		
raise right hand	10		
raise both hand	10		
TBD	10		

Table 1. Evaluation table for pose2action conversion

Finally, in order to evaluate the overall performance, we'll provide a Google form for player to rank for us after playing.

3. Milestones

3.1. StyleGAN

We have done the style encoder part and the result are shown in Figure 3. The left part of image is the original image while the right part of image is the reconstructed image. The reconstruct image's eyes, eyebrow, nose and mouth is quite different from the original image. We find some potential problems about it:

1. The output latent code from style encoder is not the best suited latent code. Maybe we can retrain the model or tune the hyperparameters.

2. Unfortunately, there is the best output from the style encoder model trained on FFHQ dataset [4]. Maybe we can try other state-of-the-art encoder or train on different dataset (CelebaHQ [3]).

3.2. Human pose estimation

We have already done the real-time camera capture and can predict the estimation of 33 key points of each person in high frequency per second. We get the predicted position x and y of each key point from both people so

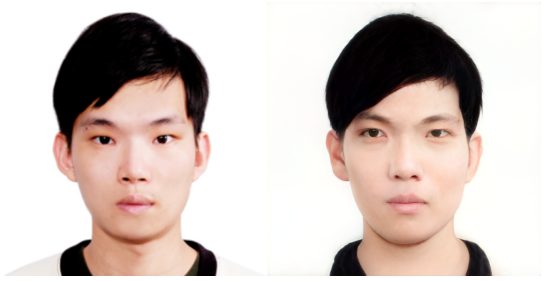

Figure 3. Style Encoder result (left: original image, right: reconstructed image).

that we make the human pose estimation matrix $M_{left} = [[x_1, y_1], [x_2, y_2], ..., [x_{33}, y_{33}]]$ for the left side person and $M_{right} = [[x_1, y_1], [x_2, y_2], ..., [x_{33}, y_{33}]]$ for the right side person. Figure 4 shows the predicted results of key points when we let two players stand in front of the camera.

Figure 4. The 33 key points of two people.

3.3. Game interface

Our game interface part refers to this project [1]. We have already modified and implemented the basic functions, including the establishment of game logic, the display of the screen, and the control of the fighters with the keyboard. The result so far shows in Figure 5.

In the game logic part, the character (stickman) has the following basic actions: move left and right, jump up, attack. The moving range of the character does not exceed the game window; the jump height is limited by gravity, and the attack has a certain judgment range. When the enemy is within the judgment range, the HP will be reduced. When one party's health reaches zero, the game ends.

The part of the screen display includes the game background, characters and life bar. For the part of the character display, we draw the sprite of the stickman. The sprite is composed of many sub-pictures, and a sequence of sub-pictures in a certain row will correspond to a certain action.

For example, the action of attacking will correspond to a sequence, which is composed of 8 sub-pictures. After receiving an action command, the corresponding pictures in the picture sequence are displayed in sequence, forming the animation in the game window.

In the keyboard control part, each action of the character will correspond to a keyboard, such as left: `A`, right: `D`, jump: `W`, attack: `R`, `T`.

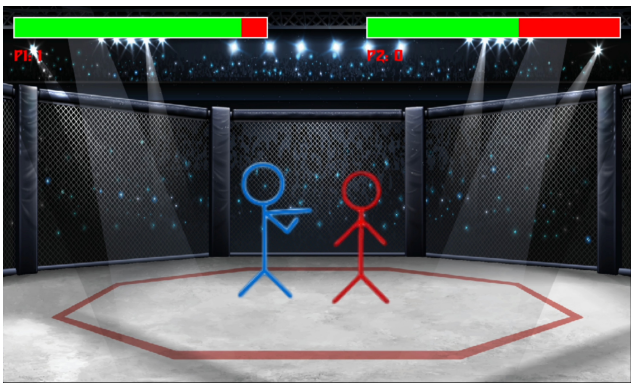

Figure 5. The game GUI.

4. Remaining milestones

For styleGAN, we need to come up with a better way to find latent code which can reconstruct the human face as similar as possible. We expect that if we have a better latent representation, then the interpolation result will also be better.

For human pose estimation, since we can already capture the key points of human, so the remaining part will only be the Pose2Action, which will classify the key points movement into some pre-defined actions.

For game interface, we need to integrate with the previous two parts. First, the face of the stickman will change to the output of the styleGAN images, and switch them sequentially according to the HP of the character. Second, the action control of the stickman will change from the keyboard input to the output of Pose2Action. In another word, player can use his/her body movement to control the game character.

Following are some checkpoints:

- Checkpoint 1: Finish Pose2Action and integrate with game interface. (by 12/06)

- Checkpoint 2: Finish StyleGAN and integrate with game interface. (by 12/09)

- Checkpoint 3: Finish overall testing, the whole pipeline should work properly. (by 12/16)

The schedule is shown in Figure 6.

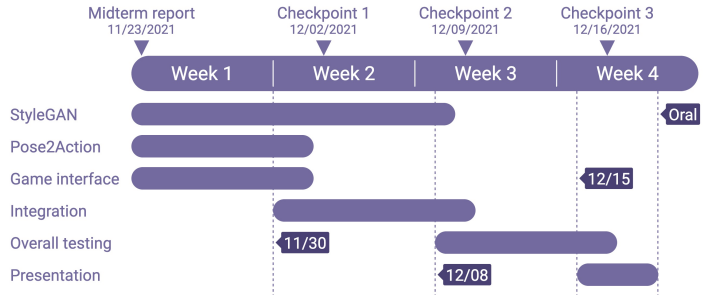

Figure 6. Gantt chart for our schedule.

References

[1] https://github.com/russs123/brawler_tut. 3

[2] Camillo et al. Mediapipe: A framework for building perception pipeline. https://github.com/google/mediapipe, 2019. 1

[3] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 3

[4] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 1, 3

[5] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020. 1

[6] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2287–2296, 2021. 1

[7] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. 2022. 1