ual: creative computing institute

Creative Making: MSc Advanced Project

University of the Arts London

Creative Computing Institute

Crossing the latent space

Real-Time Gestural Interaction with Deep Image Generative Model

---

Author:

Nuo Liu

22019706

Supervisor:

Jasper Zheng

December 1, 2023

Abstract

Use gestures as inputs to control deep image generative models, creating real-time interpolated transitions. Gesture recognition enables computers to understand human body language, filling a gap in textual or traditional graphical interfaces. This paper explores using hand gestures as inputs for a deep image generative model (StyleGAN3). The spatial position of the hand is mapped in real time to the latent space of the model, and images of the associated latent vectors are generated from the data captured by the camera. Mapping real-time gesture data input to latent vectors, then generating the corresponding latent vector image in real-time and rendering it to the interface, and the corresponding latent vector-generated image can be saved by the pinch gesture at the time of operation.

Acknowledgments

Thanks to my Supervisor japser for his guidance and help, my friend Bo Shui for asking me all sorts of basic questions, and my study partner Liao Runyi for her help.

Contents

1 INTRODUCTION

How can we interact with the ML model? The ChatGPT product gives a great way of interacting with NLP models - conversationally, it's quite like our digital chats, which is one of the reasons why we can use it so easily. With the development of ML models, many products based on ML models are emerging as tools to assist people's work and life, so how the public can simply use the models will become a very worthwhile direction to explore.

The most basic ML model of interaction is the command line [9,10,11], through command codes as input.

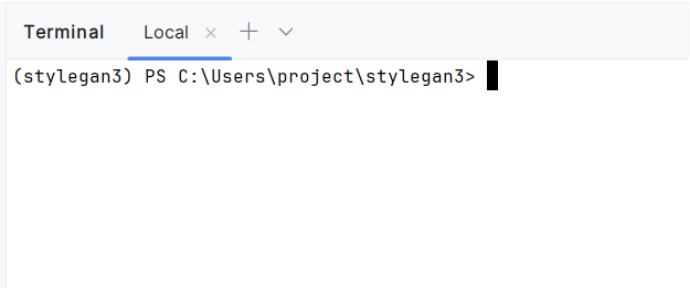

Figure 1: **Command Line Interface (CLI).**

Due to the high expertise required by command-line interaction, many developers have begun to build graphical user interfaces (GUI) for ease of use. Most current ML model GUI interactions provide a parameter panel that supports numerical adjustment of parameters to control the model output [3,7]. Visualizer that adapts the command line into a window with simple UI components for value modification and buttons. Examples include Stable Diffusion's model GUI interface, and DragGAN uses an Interactive Point-based interface to support point-based modifications to the image. [3]

Figure 2: **The DragGAN model GUI interface.**

For the same type of modal transformation ML model, the image-to-image ML model supports painting for (real-time) interactive generation, which is more useful for generating the requested

images.[2]

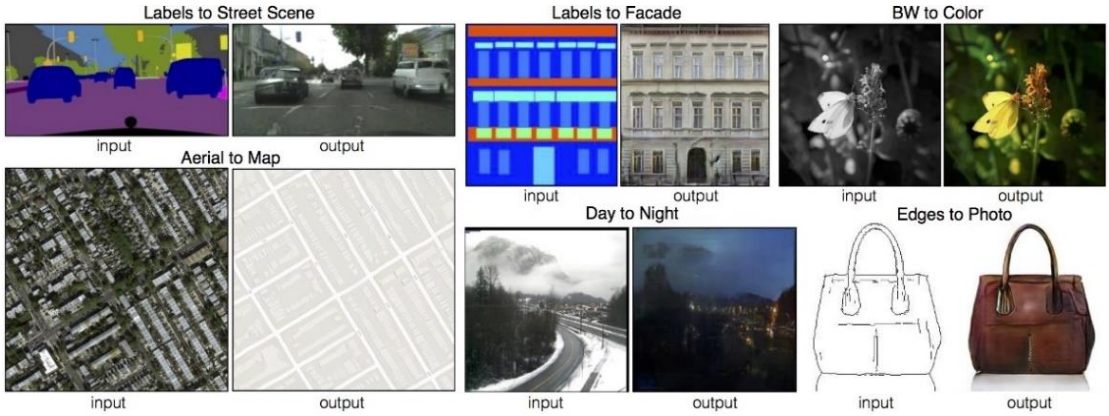

Figure 3: **Image-to-Image Translation with Conditional Adversarial Networks.**

As developers of ML models optimize interactions to simplify the instrumental use of the models, many artists are exploring novel ways of interacting with ML models. “Learning to See: Interactive” [1]is an interactive installation in which a number of neural networks analyze a live camera feed pointing at a table covered in everyday objects. Through a very tactile, hands-on experience, the audience can manipulate the objects on the table with their hands, and see corresponding scenery emerging on the display, in real-time, reinterpreted by the neural networks. The interaction can be a very short, quick, playful experience.

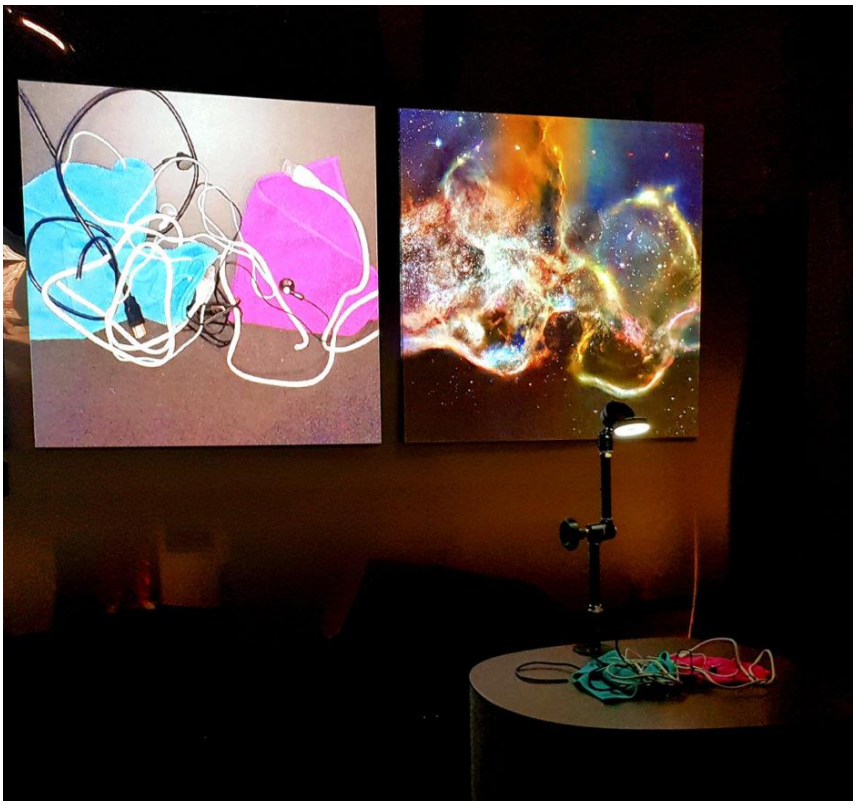

Figure 4: **Installation view at “AI: More than Human”, The Barbican, London, 2019.**

Using the camera to capture the actual living space transmission to the ML model to generate images. Technically, this work belongs to image-to-image generation. What sets this work apart is that the space for interaction is the actual physical space, which can allow for real-time interaction.

1.1 Research Goals

The goal of my research is to use gesture inputs as control interactions using the deep image generative model to generate interpolated transitions in real-time.

The use of gesture interaction[6] can break away from the traditional GUI interface interaction and use the actual physical space as a three-dimensional interaction space. Also, hand interaction in real space can map the latent vector space to real space, which helps to contribute to the understanding of latent space in machine learning, providing the layman with a better understanding of the principles of machine learning.

Being able to run and interact in real time is a very high mark and requirement, which requires a very high frame rate requirement, as well as a single processing speed that is fast enough to achieve a real-time display.

2 Related Work

2.1 X Degrees of Separation - Mario Klingemann

X Degrees of Separation[12]is an installation and online artistic experiment made in collaboration with Google and artist Mario Klingemann. Using machine learning techniques that analyze the visual features of artworks, X Degrees of Separation finds pathways between any two artifacts, connecting the two through a chain of artworks. This network of connected artworks takes us on a scenic route where serendipity is waiting at every step.

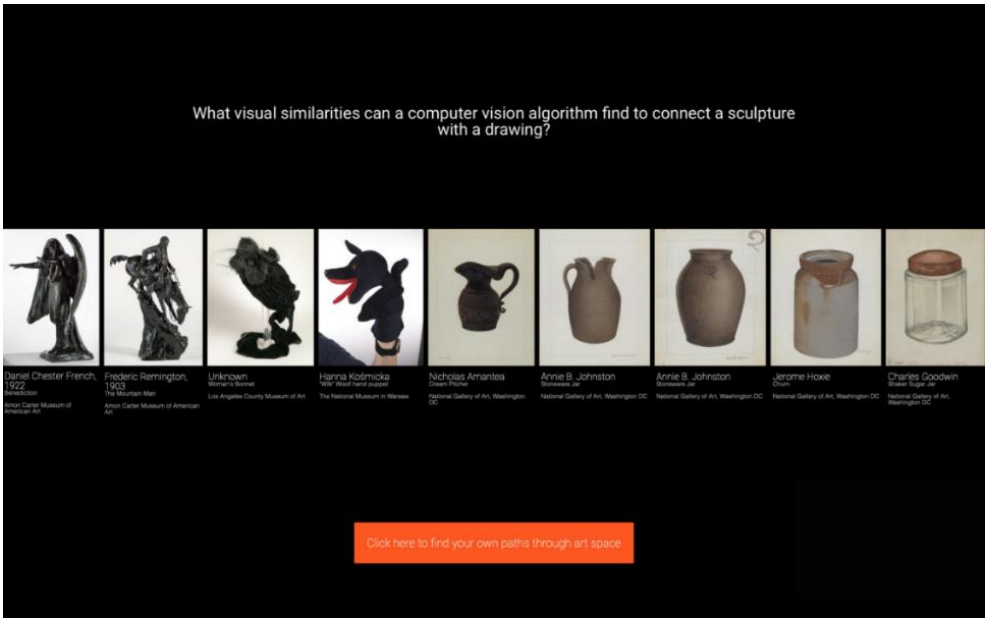

Figure 5: **X Degrees of Separation by Mario Klingeman**.

This work uses many photographs of artworks as a database, using deep learning-based techniques such as Generative Adversarial Networks (GANs) or Variable Auto-Encoders (VAEs). These models learn the features and content of the images and classify them in a latent vector space. Using path search algorithms, the intermediate images of the path are made to visually form a natural transition or leap, so that each intermediate image is associated with the preceding and following images.

2.2 Latent space

A latent space[13], also known as a latent feature space or embedding space, is an embedding of a set of items within a manifold in which items resembling each other are positioned closer to one another. Position within the latent space can be viewed as being defined by a set of latent variables that emerge from the resemblances of the objects.

In most cases, the dimensionality of the latent space is chosen to be lower than the dimensionality of the feature space from which the data points are drawn, making the construction of a latent space an example of dimensionality reduction, which can also be viewed as a form of data compression. Latent spaces are usually fit via machine learning, and they can be used as feature spaces in machine learning models.

The interpretation of the latent spaces of machine learning models is an active field of study, but latent space interpretation is difficult to achieve. Due to the black-box nature of machine learning models, the latent space may be completely unintuitive. Additionally, the latent space may be high-dimensional, complex, and nonlinear, which may add to the difficulty of interpretation. Some visualization techniques have been developed to connect the latent space to the visual world, but there is often not a direct connection between the latent space interpretation and the model itself. Such techniques include t-distributed stochastic neighbor embedding (t-SNE), where the latent space is mapped to two dimensions for visualization. Latent space distances lack physical units, so the interpretation of these distances may depend on the application.

2.3 Gesture Recognition

Gesture recognition is an area of research and development in computer science and language technology concerned with the recognition and interpretation of human gestures. A subdiscipline of computer vision, it employs mathematical algorithms to interpret gestures. Users can make simple gestures to control or interact with devices without physically touching them. Many approaches have been made using cameras and computer vision algorithms to interpret sign language, however, the identification and recognition of posture, gait, proxemics, and human behaviors is also the subject of gesture recognition techniques. Gesture recognition is a path for computers to begin to better understand and interpret human body language, previously not possible through text or unenhanced graphical user interfaces (GUI). I found that Gesture recognition falls into two main clusters, which each inspired my study in different ways:

(Sensor-Based Detection) Leap Motion Controller: The Leap Motion Controller is a small USB peripheral device that is designed to be placed on a physical desktop, facing upward. Using two monochromatic IR cameras and three infrared LEDs, the device observes a roughly hemispherical area, to a distance of about 1 meter.

Figure 6: **Leap Motion Controller**.

Leap Motion is a hand-capture product with two built-in cameras, the efficiency and accuracy of the capture are the best among many ways, with relative SDK software and unity/unreal plugin, easy to use. A variety of art programming software also supports Leap Motion. Due to Leap Motion's previous products, the related tutorials are very old and have low references due to software adaptability and other reasons, and only the official updated documents are available. The old SDK V2.0 supported C++ and switched to a multi-programming language API, which is not supported in the current Gemini version.

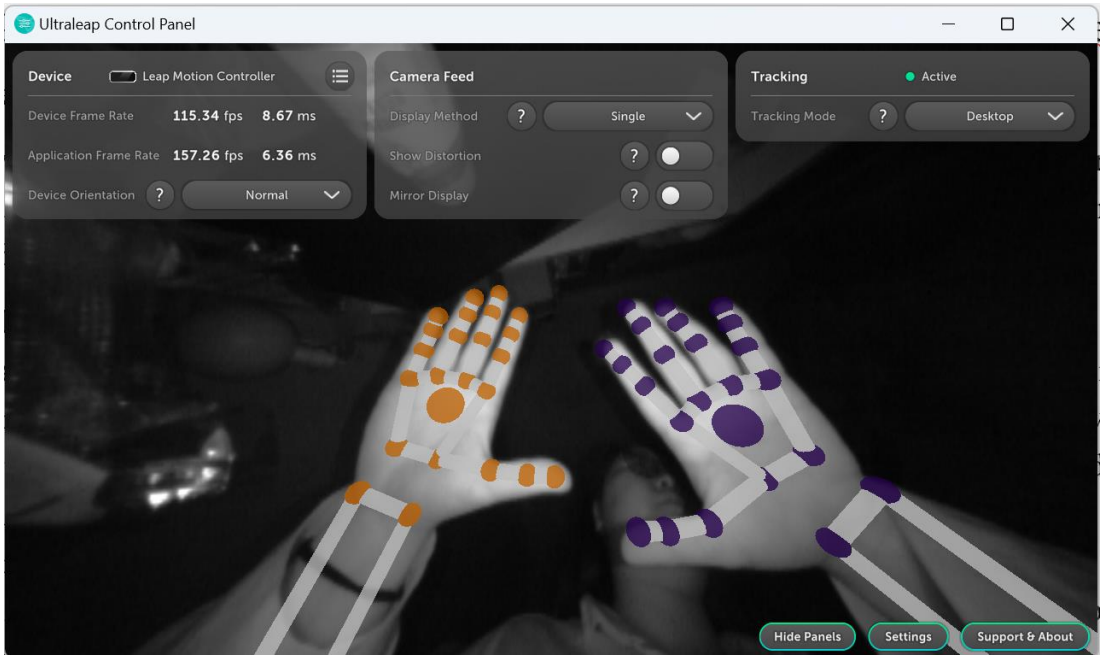

Figure 7: **Leap Motion Control Panel (Gemini Ver.).**

(Hand Tracking Module) Computer Vision, Gesture Recognition Libraries and APIs: Creating a hand tracking module [4] using Python, OpenCV, and MediaPipe. OpenCV, a powerful computer vision library, helps in processing the video input and performing tasks like image preprocessing, contour analysis, and feature extraction. It handles the visual data captured by the camera, enabling the identification and extraction of relevant information related to hand positions and movements. MediaPipe utilizes machine learning techniques, specifically deep learning models, to recognize and track hands accurately to understand and predict hand gestures and movements. It simplifies the implementation of hand tracking by providing pre-built models and tools for developers to easily integrate into their applications.

Build through Hand Tracking Module will have much more flexibility to adjust the capture as needed without restrictions of the SDK black box. Compared to Leap Motion's two built-in cameras, OpenCV's accuracy drops sharply by using a single camera. For example, Leap Motion can detect the XYZ three-axis, while OpenCV supports two-dimensional detection, which decreases the accuracy in one dimension.

Table 1: **Equipment Comparison**

Equipment	Software/library	Detection accuracy	support language
Leap Motion	SDK	three-dimensional	C++
Camera	OpenCV/MediaPipe	two-dimensional	Python

2.4 StyleGAN3

StyleGAN3[7] is an evolution of the StyleGAN series, a type of generative adversarial network (GAN) developed by researchers at NVIDIA. It aims to generate high-quality and realistic images by learning from a large dataset of images and creating new ones with similar characteristics. StyleGAN3 is equipped with an interactive image generation GUI interface (imgui) and real-time rendering (openGL) functionality, which supports functions such as parametric adjustment of image generation and mouse-controlled seed to produce image changes.

3 Methodology

The current models for image generation are dominated by two main classes: stable diffusion (SD) and generative adversarial network (GAN). Compared to SD, the GAN model generates images faster and can reach the standard of real-time generation [8], meanwhile, the image generation is more controllable. The main development of this project is based on the StyleGAN3 model and its visualizer. The focus is on the interaction with gestures as inputs, database tuning, and model training are not the focus of concern, so use the StyleGAN3 pre-trained models given officially by StyleGAN3.

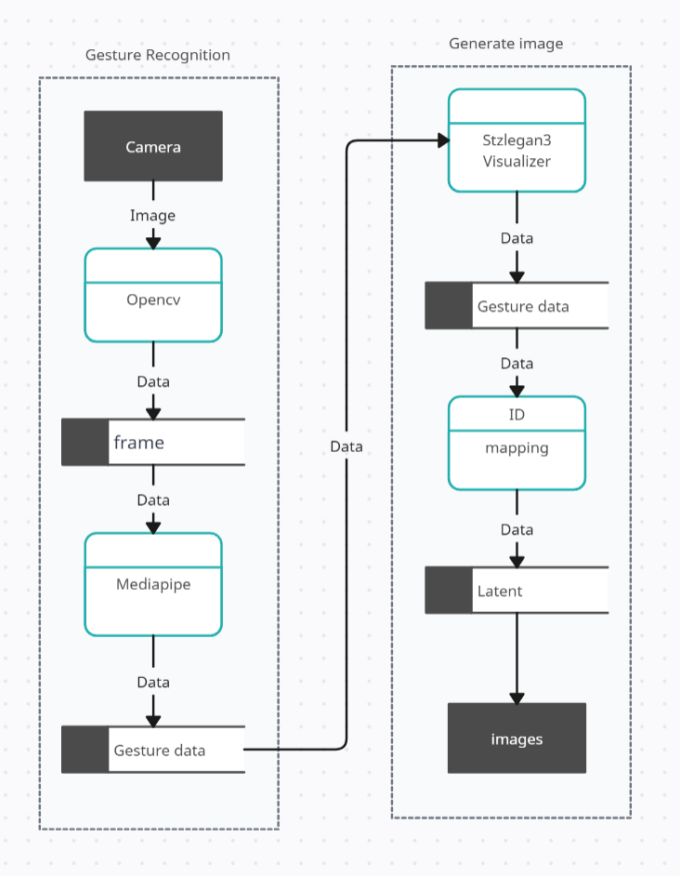

Figure 8: **Data Flow Diagram**.

By receiving the gesture data, it is possible to relate the spatial position of the hand to the vector values of the latent vectors in the model. We can get the positions of the two hands to generate two latent vectors, and then linearly interpolate these two latent vectors to help people have a better comprehension of the principle of the latent space. To break down the overall development, we need to be able to capture gestures, generate interpolated images in the StyleGAN3 model, and then combine these.

4 Project Process

4.1 Smooth path in StyleGAN3

In StyleGAN3, smoothing the path refers to controlling the interpolation between two latent vectors to generate smoother transitions in the generated images. This process allows for gradual changes between latent vectors, resulting in more continuous and visually appealing transformations.

During this phase, the linear interpolation[14] formula to achieve a smooth transition of the generated image is as follows:

```
def lerp (u, v, a):
  return a * u + (1 - a) * v
```

Combine the linear interpolation function with StyleGAN3's original image generation gen_images.py file. The image smoothing transition is generated by inputting the parameters seed_a, seed_b, num_steps, to determine the two vector End Points and Steps.

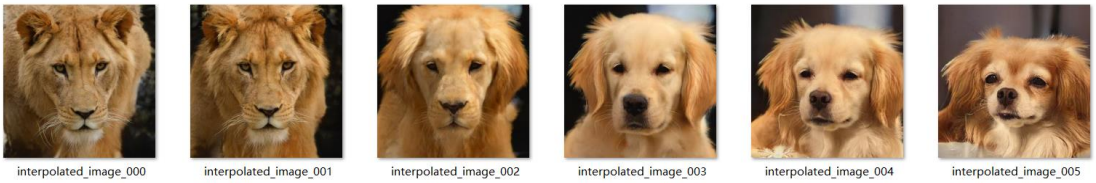

Figure 9: **Linear interpolation series image generation(step=5)**.

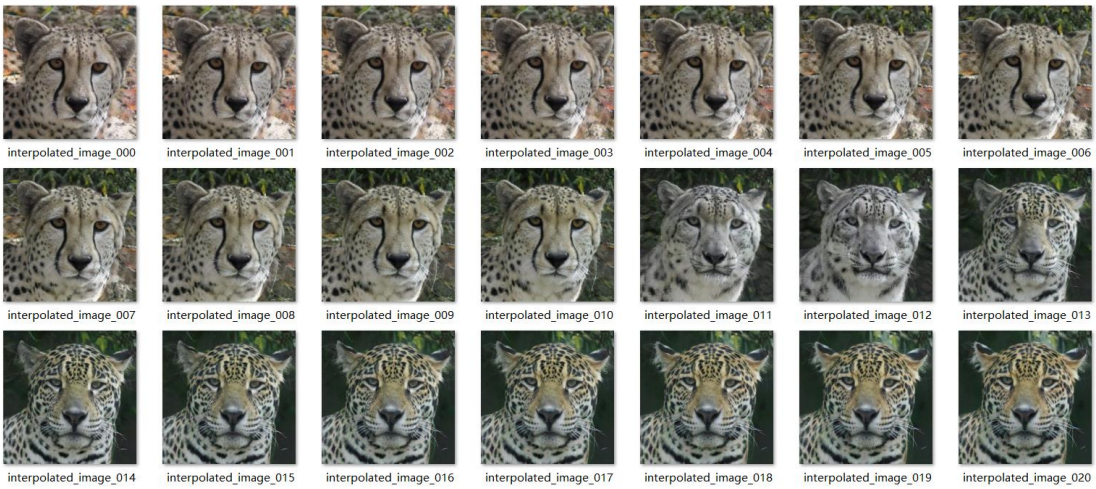

Figure 10: **Linear interpolation series image generation(step=20).**

The difference between the images is related to the distance between the two endpoints of the interpolation and the step size. The larger the step, the more interpolated images are generated and the smaller the difference between the images.
In addition, performing the display of smooth transitions can enhance the understanding of the methods of feature classifying in machine learning.

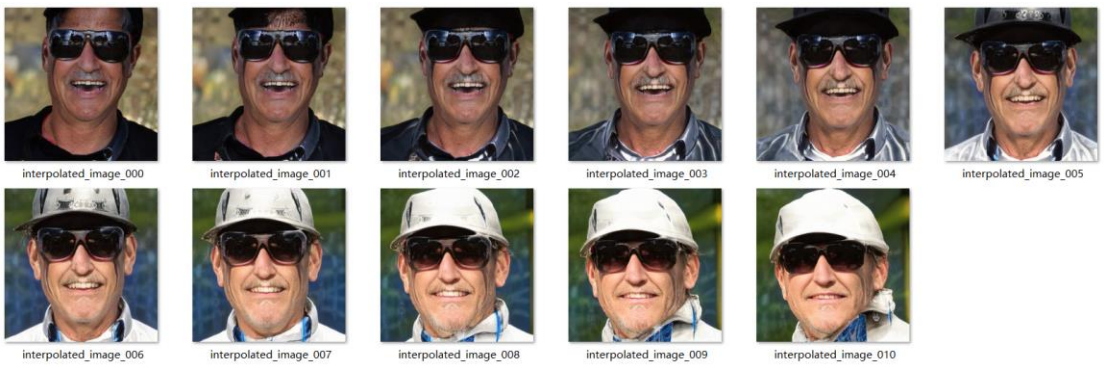

Figure 11: **Linear interpolation series image generation(step=10).**

The smooth transition in the Figure 11 identifies the side shadows of the face and the shadow

of the sunglasses on the face as similar features, which can show the difference between how the machine operates and how the human mind understands images.

4.2 Gesture recognition with Leap Motion and OpenCV & Mediapipe

1） Leap Motion.

Firstly, try to use Leap Motion for gesture capture because it has two in-built cameras and well-developed spatial computation algorithms to acquire highly accurate three-dimensional hand data.

The SDK for leap motion supports unity and unreal development, but I need to bridge it with StyleGAN3's model via the Python language. The SDK for leap motion is based on .lib files written in C++. In the old V2.0 version of the SDK, there were multi-language APIs available to retrieve data from the controller, but now multi-language APIs have been removed in the current version (Gemini) of the SDK. To achieve a better result, I tried to build the SDK's Python language API interface, the tutorial Generating a Python 3.3.0 Wrapper with SWIG 2.0.9 [5]in the Ultraleap developer forums was released in 2019, I attempted and ended up with a failure.

Leap Motion can also get the gesture data through the visual node programming software TouchDesigner (TD), then use TD will export the gesture data to a .csv file, StyleGAN3 model through the python language to get the local .csv file data way to get the gesture data.[15]

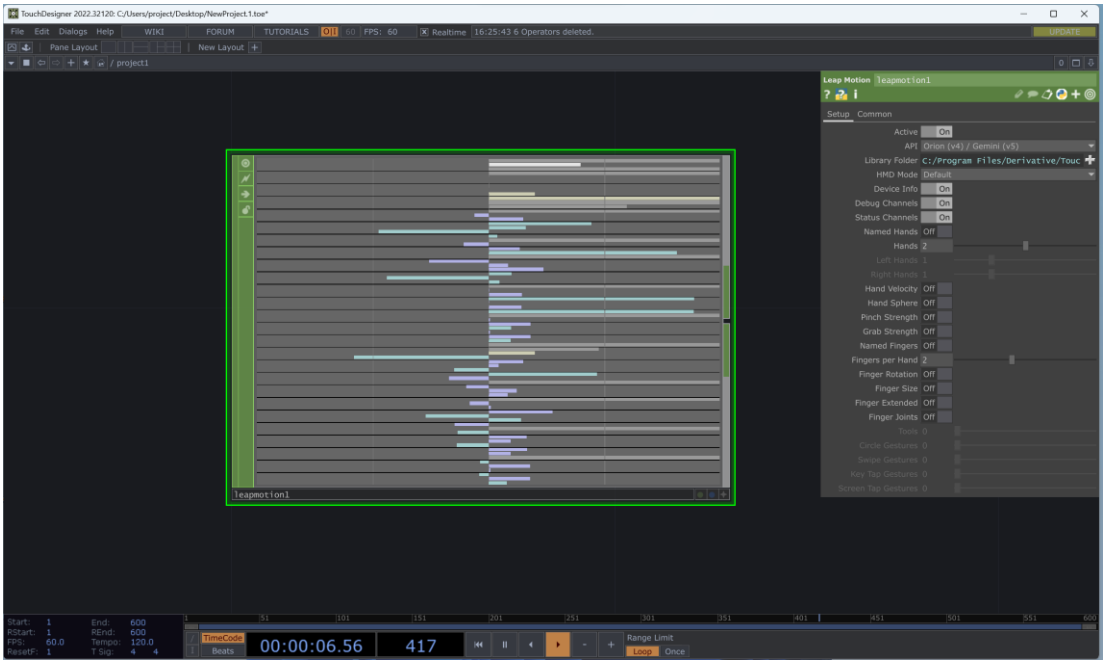

Figure 12: **TD Leap Motion Component.**

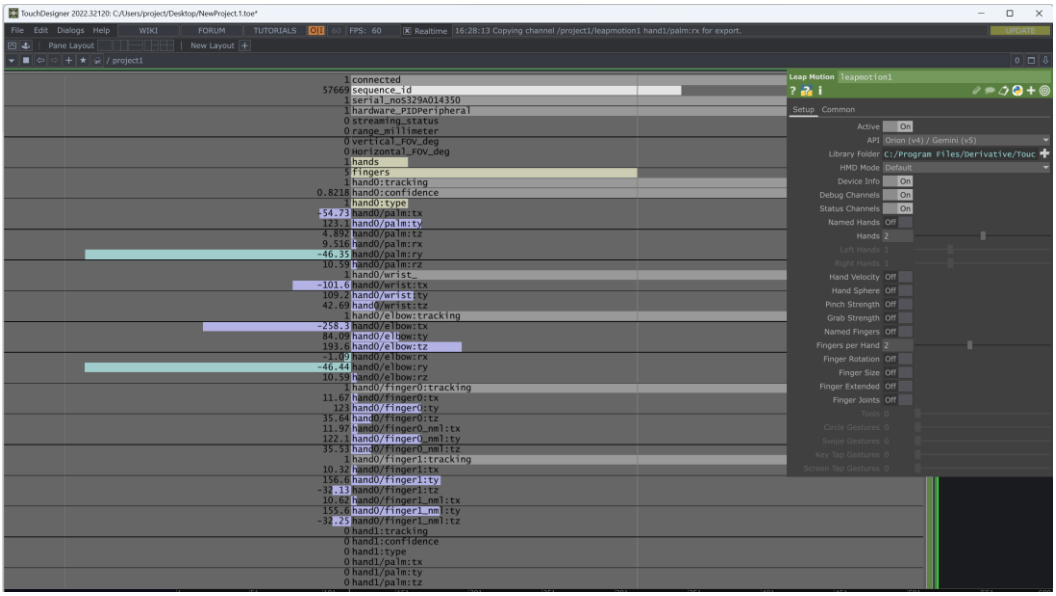

Figure 13: **TD Leap Motion Data.**

Real-time requires continuous refreshing to get data. The refresh rate Leap Motion uses in TD allows for real-time results. Data transmission through TD as a third-party software cannot achieve real-time function, and there will be a data overflow phenomenon.

2） OpenCV & Mediapipe.

Gesture recognition using computer vision through the camera is in Python and is compatible with the StyleGAN3 model environment, supporting real-time access to hand data.

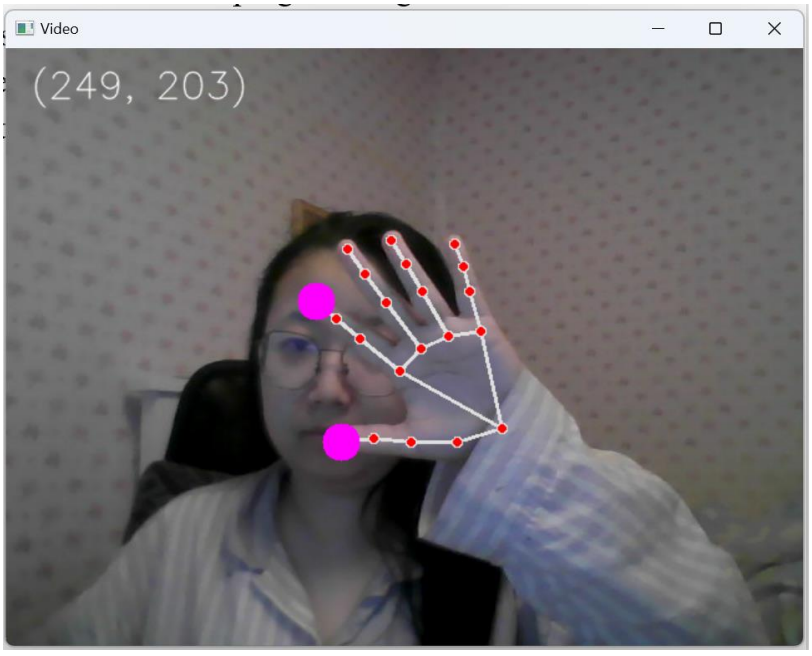

Figure 14: **OpenCV&Mediapipe.**

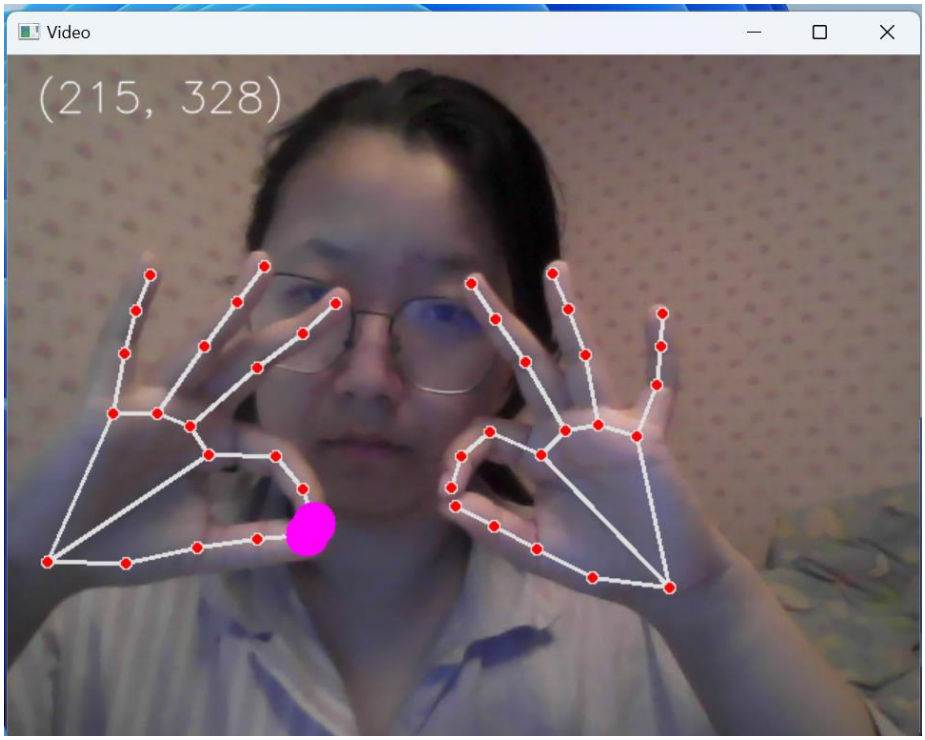

Figure 15: **Versions of two-handed testing.**

I use the detection of pinch gestures, i.e., detecting if the thumb and index finger are close together, So the positional distance between the thumb and index finger vertices is detected in real time, and when the distance is close, it is judged to be a pinch gesture (reserved for the scope of accuracy). In actual testing, it was found that a single pinch gesture would be reported continuously, in order to optimize the code, it was set that the pinch before a non-pinch gesture would only be reported once.

```
if thumb_tip is not None and index_tip is not None:
    distance = math.sqrt((thumb_tip[0] - index_tip[0]) ** 2 + (thumb_tip[1] - index_tip[1]) ** 2)
    if distance <= 20:
        if self.set_latent_widget(self.latent_widget):
            print(f"pinch detected")
def set_latent_widget(self, latent_widget):
    if self.last_latent_widget != latent_widget and latent_widget is not None:
        self.last_latent_widget = latent_widget
        self.latent_widget = latent_widget
        return True
    return False
```

In the actual practice, there exists the situation of frame dropping and lagging, and at the same time, there are fewer errors in the capturing of gestures, but on the overall real-time gesture capture impact is within the acceptable range.

4.3 Mapping hand data to latent

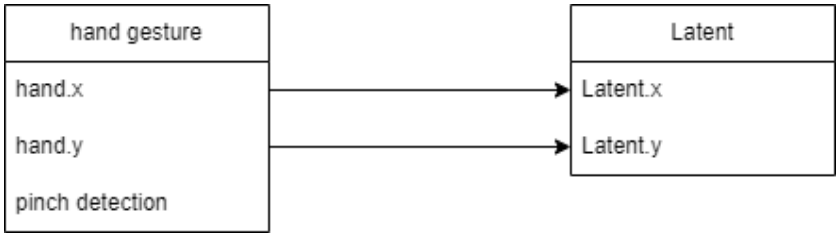

Figure 16: **Mapping hand data to latent**

Using gesture capture to get the 2D data of X and Y of the hand's position, pass the data to the x and y properties of the latent through the function drag. Adjusting the sensitivity of the gesture to the influence of potential vectors by a scaling factor (viz.font_size * 4e-1).

```
def drag(self, dx, dy):
    viz = self.viz
    self.latent.x = dx / viz.font_size * 4e-1
    self.latent.y = dy / viz.font_size * 4e-1
```

When the position of the hand changes, the relative latent assignment is adjusted to modify the resulting image.

4.4 Visualizer

The StyleGAN3 model visualizer is an encapsulated system. It is divided into two main parts: parametric adjustment of the model and image rendering. There is a very large amount of packaged content imported into visualizer to support all sorts of complex parameter tuning.

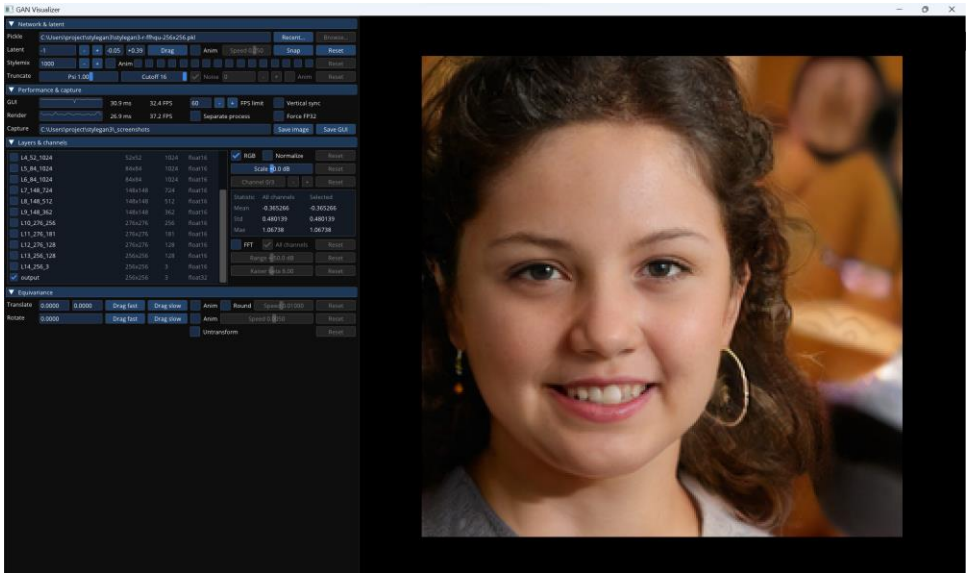

Figure 17: **The StyleGAN3 model visualizer.**

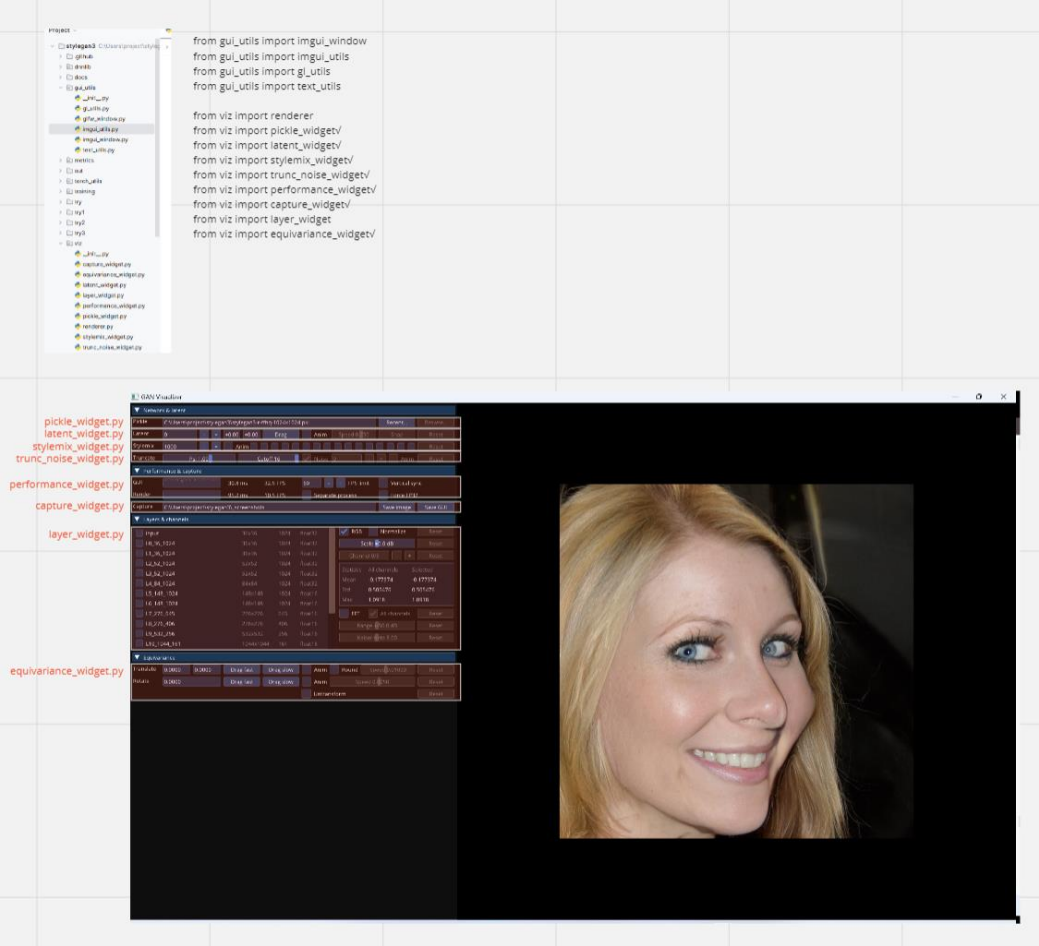

Figure 18: **Notes on understanding visualizer's import file.**

I only need to use latent parameter modification when I actually use it, so I need to unpack the functions that I didn't use and simplify the code and functionality by normalizing the parameters and using default values for this part of the functionality.

Figure 19: **Adjusted interface.**

I removed all parameter adjustments except latent_widget and removed the panel part of the interface to get an interface that allows for real-time latent adjustments and rendering of the generated image. The adjusted visualizer interface is shown in Figure 19.

5 Result: Gesture-driven model control

In order to control the image generation of the model through hand data, gesture detection is added to the code that generates the image, and the main code logic is shown in Figure. 20 below, and the latent is mapped and associated with the hand data, so that the generated image follows the hand in real time.

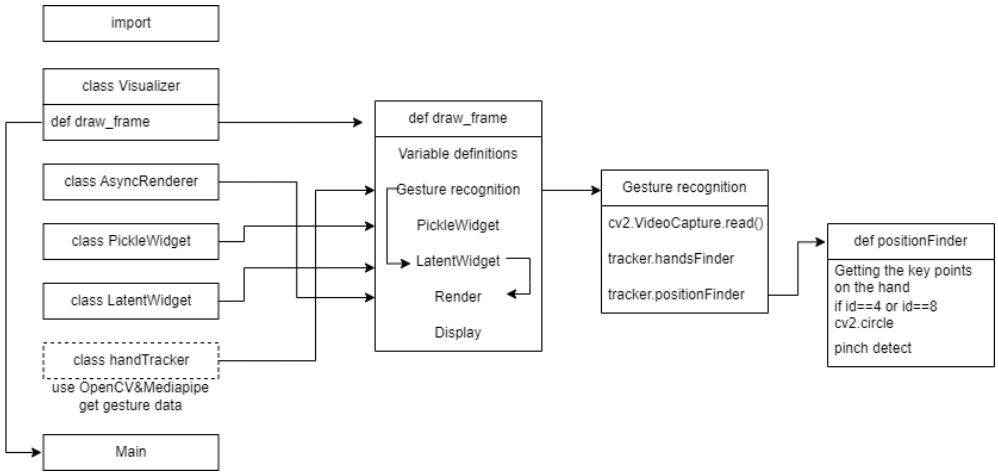

Figure 20: **The main code logic for real-time image generation.**

With the above logic, single frame gesture recognition is added to the function of image loop generation so that the whole process of recognition to generation is done as a single loop part, which requires faster processing speed to complete the effect in real time.

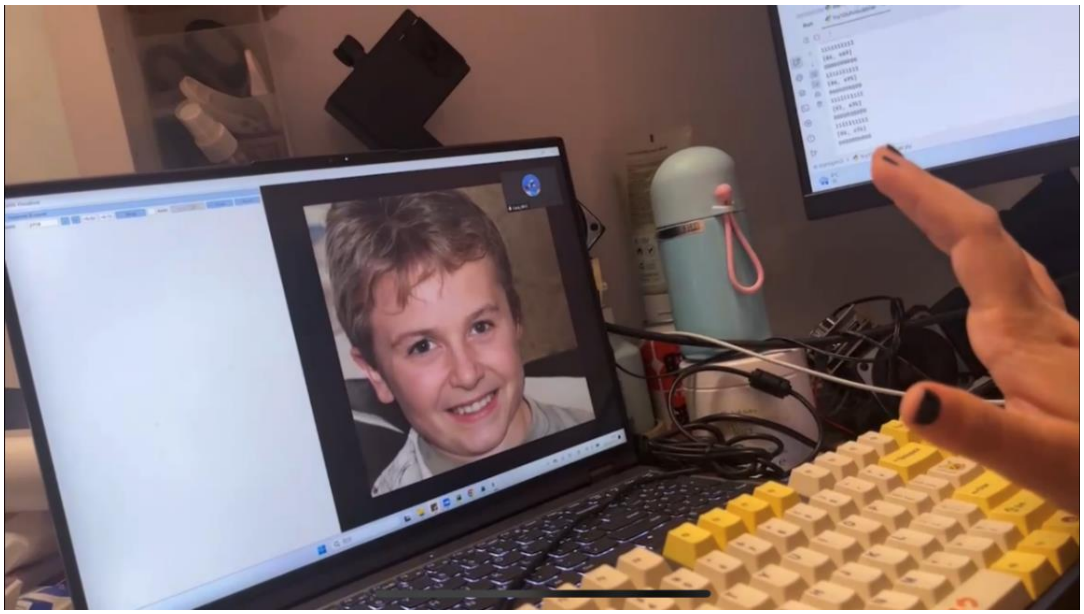

Figure 21: **Screenshots of actual operation demo.**

In the basic demo version, the gesture data is printed out in the console while the images in the interface are generated in real time based on the binding of the surgical data to the latent vectors. Since there is no real-time feedback of the gesture capture effect, there will be a situation where the hand cannot be effectively captured outside the camera, so the captured image of the camera should be displayed so that the user can adjust the hand interaction position to improve the effective capture situation. An iterative version of the interface that takes the interface to camera content display is shown in Figure 22 and Figure 23 below.

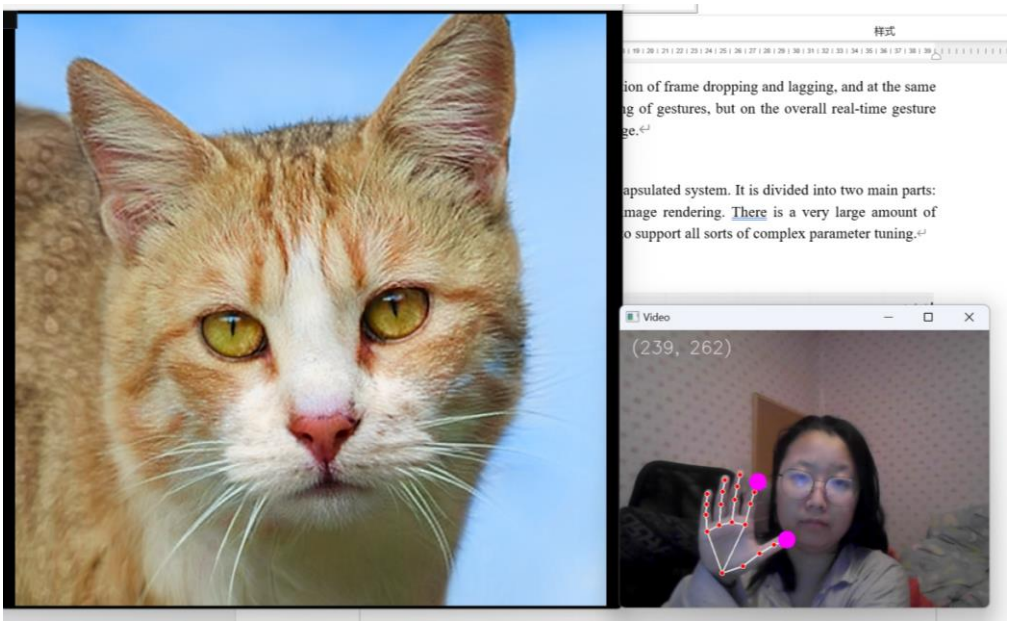

Figure 22: **Screenshots of Interface Iteration version 1.0 (hand data (239,262)).**

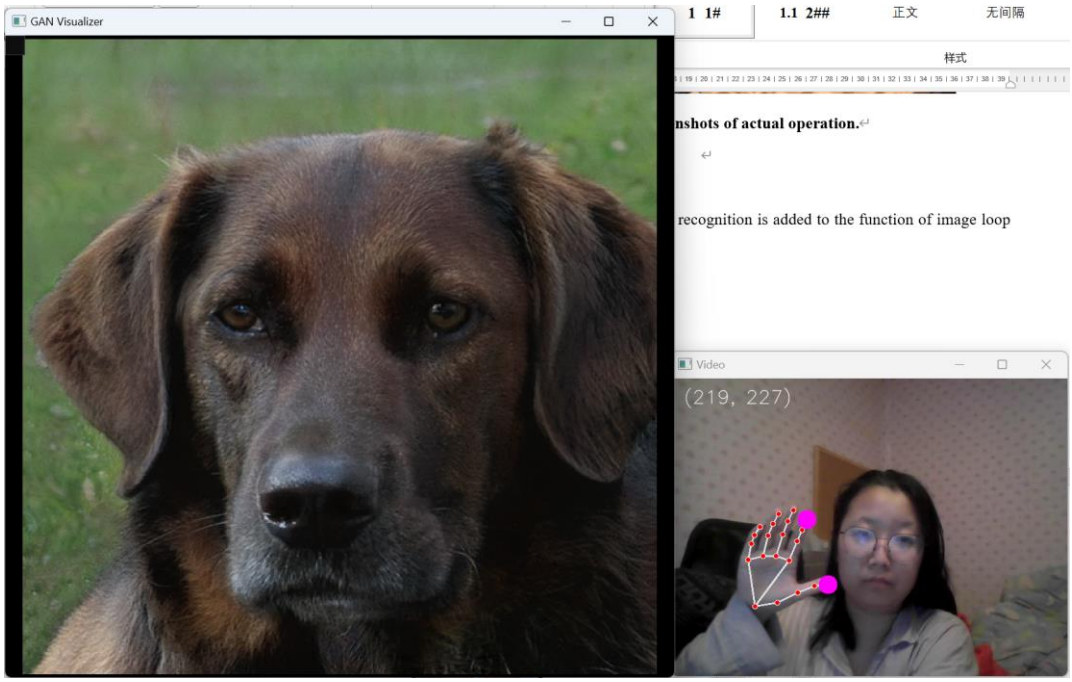

Figure 23: **Screenshots of Interface Iteration version 1.0 (hand data (219,227)).**

When the user puts the hand into the interaction area, an image of the corresponding latent position is generated based on the position of the hand. When the hand position moves, the position of the latent changes. Since the movement of the hand position is a continuous process, the generated image will also undergo a continuous change. The image is continuously smoothed throughout the user's interaction.

Support saving generated image to local file by gesture pinch. When the user makes a gesture pinch, the gain current latent data is transferred to the image local generation function for local storage. Then local images can be accessed by a third party to forward user-generated images to online support for user access so that users can get the generated images they want.

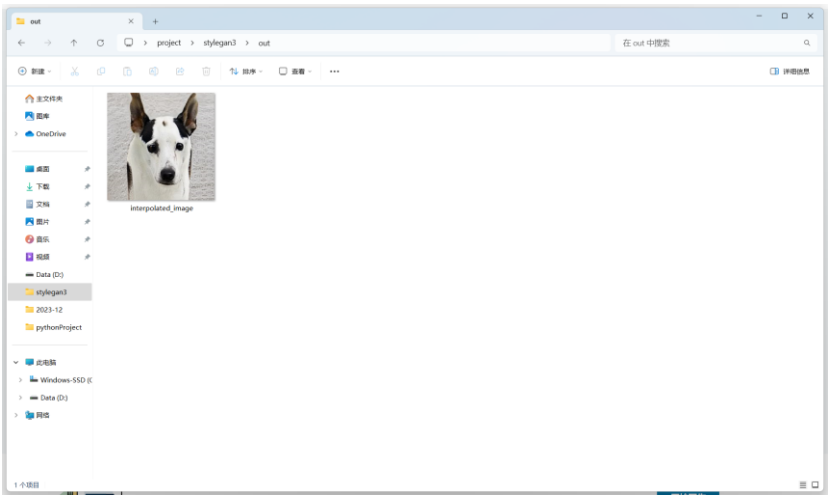

Figure 24: **Generate images for local storage.**

6 Discussion

Using gestures as input for ML model interaction transfers the manipulation area of the model from the screen to real space. Undoubtedly, this kind of natural somatosensory-based interaction is very novel and can give people an interesting experience. Generating images through gesture pinch downloads is also a very creative idea. Although slightly different from the expected scenario, it still managed to experiment with gestures as a way of interacting with models in real-time.

Gesture-based interactions can be very simple to control in three dimensions, one more dimension than on-screen mouse interactions. Closer to the way people naturally interact with physical language hand movements and gestures can be combined on the design to reduce the learning cost of use.

However, due to technical limitations, gesture interaction cannot complete complex interactions, as a result not recommended as a mainstream interaction method. We were unable to obtain gesture data with high accuracy and low capture error rate. This is because the current recognition methods for gesture interaction are mainly based on computer vision, removing the problem of dealing with very low recognition errors of models, which are currently captured by camera-based sensors. When capturing gestures with a camera as a sensor, it is difficult for a single camera to create a three-dimensional concept, thus losing the advantage of gesture capture. Sensors captured by multiple cameras are often integrated sensor devices with varying degrees of restriction on the freedom of use.

Due to the development time and my personal code skill level, many of the design parts of the development that did not get done in time for the work will be described and demonstrated to show the final result and shape of this project.

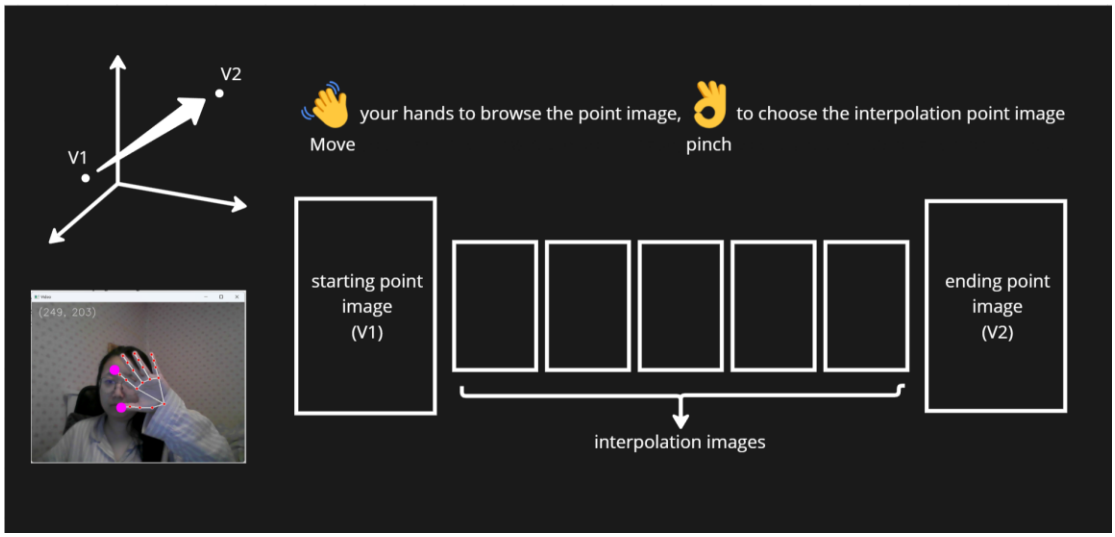

Figure 25: **GUI Design**.

The interface of the final work is shown in Figure 25. In the interface there is a simple guide to the interaction process, and below it is an area for linearly interpolated image generation, which

supports two-handed operation. The spatial positions of the left and right hands are assigned to the left and right endpoints of the linear interpolation, and an interpolated image is generated in the middle of the pinch; when the distance between the hands increases during the pinch, the number of interpolated images is reduced in steps of increasing size to see more detailed changes. The top left side downscales the potential space into three dimensions, showing the position of the hand in the potentially downscaled space, making it easier to understand the concept of potential space. The bottom left corner displays the camera's gesture capture in order to troubleshoot interaction failures due to uncaptured gestures.

Currently the association of gesture data and potential space in this work is just purely a simple mapping relationship, and if a more strongly associated connection is needed, it needs to deal with downscaling to multi-dimensional data, with more mathematical mapping relationships and logical issues.

7 Conclusion

In this work, I propose to use hand gestures as inputs to the interactive form of the deep image generative model (StyleGAN3 model) to generate interpolated transformations that relate the position of the hand to the latent space in the model and to control the generation of the image by means of the gestures. Visualizer implementation based on the StyleGAN3 model. Linear interpolation of smooth image transitions is implemented in the StyleGAN3 model for gesture recognition through the camera. Mapping real-time gesture data input to latent vectors, then generating the corresponding latent vector image in real-time and rendering it to the interface, and the corresponding latent vector-generated image can be saved by the pinch gesture at the time of operation.

Bibliography

[1] Akten, M., Fiebrink, R., & Grierson, M. (2019). Learning to see: you are what you see. In ACM SIGGRAPH 2019 Art Gallery (pp. 1-6).

[2] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125-1134).

[3] Pan, X., Tewari, A., Leimkühler, T., Liu, L., Meka, A., & Theobalt, C. (2023, July). Drag your gan: Interactive point-based manipulation on the generative image manifold. In ACM SIGGRAPH 2023 Conference Proceedings (pp. 1-11).

[4] Simon kiruri. (2022, February 11). Creating a Hand Tracking Module Using Python, OpenCv, and MediaPipe. Section. https://www.section.io/engineering-education/creating-a-hand-tracking-module/

[5] Dan Kirby. (2019, December 16). Generating a Python 3.3.0 Wrapper with SWIG 2.0.9. Ultraleap Support. https://support.leapmotion.com/hc/en-us/articles/360004362237

[6] Chung, J. J. Y., Chang, M., & Adar, E. (2021). Gestural Inputs as Control Interaction for Generative Human-AI Co-Creation. In Workshops at the International Conference on Intelligent User Interfaces (IUI).

[7] Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., & Aila, T. (2021). Alias-free generative adversarial networks. Advances in Neural Information Processing Systems, 34, 852-863.

[8] Cao, H., Tan, C., Gao, Z., Xu, Y., Chen, G., Heng, P.-A., & Li, S. Z. (2023). A Survey on Generative Diffusion Model (arXiv:2209.02646). arXiv. https://doi.org/10.48550/arXiv.2209.02646

[9] Karras, Tero, et al. *A Style-Based Generator Architecture for Generative Adversarial Networks*. arXiv:1812.04948, arXiv, 29 Mar. 2019. *arXiv.org*, https://doi.org/10.48550/arXiv.1812.04948.

[10] Richardson, Elad, et al. *Encoding in Style: A StyleGAN Encoder for Image-to-Image Translation*. arXiv:2008.00951, arXiv, 21 Apr. 2021. *arXiv.org*, https://doi.org/10.48550/arXiv.2008.00951.

[11] Esser, Patrick, et al. *Taming Transformers for High-Resolution Image Synthesis*. arXiv:2012.09841, arXiv, 23 June 2021. *arXiv.org*, https://doi.org/10.48550/arXiv.2012.09841.

[12] Mario klingemann. (2018, March). X Degrees of Separation by Mario Klingemann.

https://artsexperiments.withgoogle.com/xdegrees/

[13] ekin tiu. (2020, February 4). Understanding Latent Space in Machine Learning. https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d

[14] Linear Interpolation. (n.d.). Wiki. https://en.wikipedia.org/wiki/Linear_interpolation

[15] Leap Motion CHOP. (n.d.). https://derivative.ca/UserGuide/Leap_Motion_CHOP