Generation of Animated 3D Characters

Roozbeh Razavi - Leo Luo - Amirhossein Alimohammadi

Introduction

Our approach comprises three main phases:

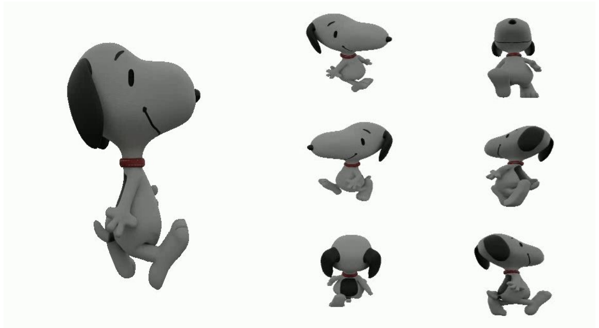
1. Multi-Frame Novel View Synthesis (MNVS): Utilizes a combination of Stable Video Diffusion and Zero123++ to generate coherent novel view videos from a single input image, incorporating both spatial and temporal priors.

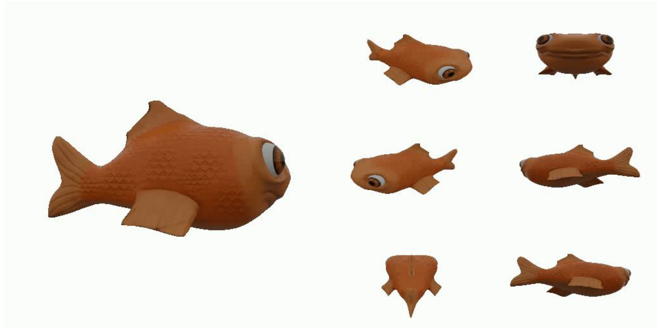
2. Score Jacobian Chaining : Explores the integration of Score Jacobian Chaining within the PAPR network to enhance its performance and robustness by leveraging the gradient information of the score function.

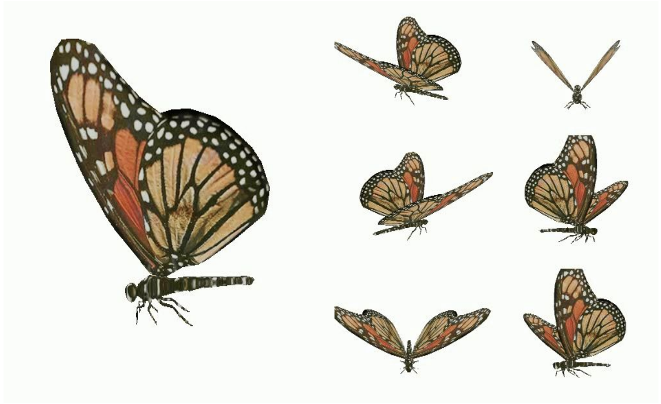
3. Generalization to Animated PAPR: Extends the PAPR framework to Animate-PAPR, enabling the generation of multi-view images for each frame of the video by training 3D asset features and renderer, and implementing a freezing and initialization strategy to propagate features across consecutive frames.

Multi-Frame Novel View Synthesis (MNVS)

First model is an image-conditioned diffusion model for generating 3D-consistent multi-view videos from a single input view.



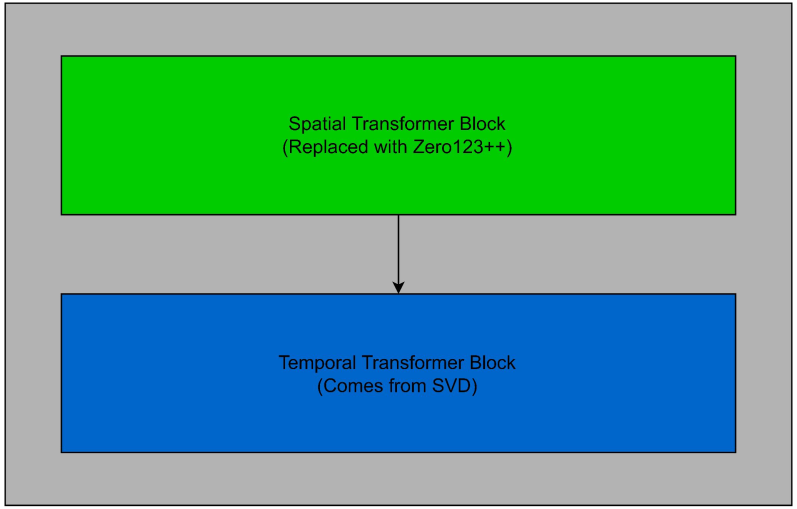




Integration of Two Models

The Multi-Frame Novel View Synthesis (MNVS) network generates consistent novel view videos from a single input image by combining elements from Stable Video Diffusion and Zero123++. This fusion allows for the incorporation of both spatial and temporal priors, resulting in accurate frame synthesis.

MNVS replaces temporal transformer blocks with those from Stable Video Diffusion and spatial transformer blocks with those from Zero123++ and fine-tunes the model using the LORA algorithm to ensure smooth integration and optimal performance.


Spatial Transformer Block
(Replaced with Zero123++)
Temporal Transformer Block
(Comes from SVD)

CrossAttn Block Spatio Temporal

FineTuning

After Combining Zero123++ and SVD, now it's time to finetune our model.

We used LORA to finetune the model and to try different parts of the model, we make LORA compatible with different modes, including,

1) Train all layers of the model
2) Train only self attention layers
3) Train q and k values only
4) Train temporal layers only

What is our Loss Function?

$$E_{t,x_0,\epsilon}\left[\| x_0 - \left(\frac{x_t}{\sqrt{\bar{\alpha}_t}} - \frac{\sqrt{1-\bar{\alpha}_t}\epsilon_\theta\left(x_t, t\right)}{\sqrt{\bar{\alpha}_t}} \right) \|_2 \right]$$

Second Phase Goal

- In this phase I tried to bring two loss function from zero123 3d reconstruction into PAPR
    - Score Jacobian Changing (SJC)
    - Near-view consistency loss
    - Input & target depth smoothing
- We are trying to learn a neural renderer
- Neural rendere should be able to synthesis image from any arbitrary view, while zero123 generate **only** limited set of views which were in Objaverse dataset
- SJC makes it possible to use those limited set of views to learn a neural renderer model by **guiding** sampling procedure

$$\nabla_{\boldsymbol{\theta}} \log \tilde{p}_\sigma(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[\nabla_{\boldsymbol{\theta}} \log p_\sigma(\boldsymbol{x}_\pi)\right] \tag{9}$$

$$\frac{\partial \log \tilde{p}_\sigma(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbb{E}_\pi \left[\frac{\partial \log p_\sigma(\boldsymbol{x}_\pi)}{\partial \boldsymbol{x}_\pi} \cdot \frac{\partial \boldsymbol{x}_\pi}{\partial \boldsymbol{\theta}}\right] \tag{10}$$

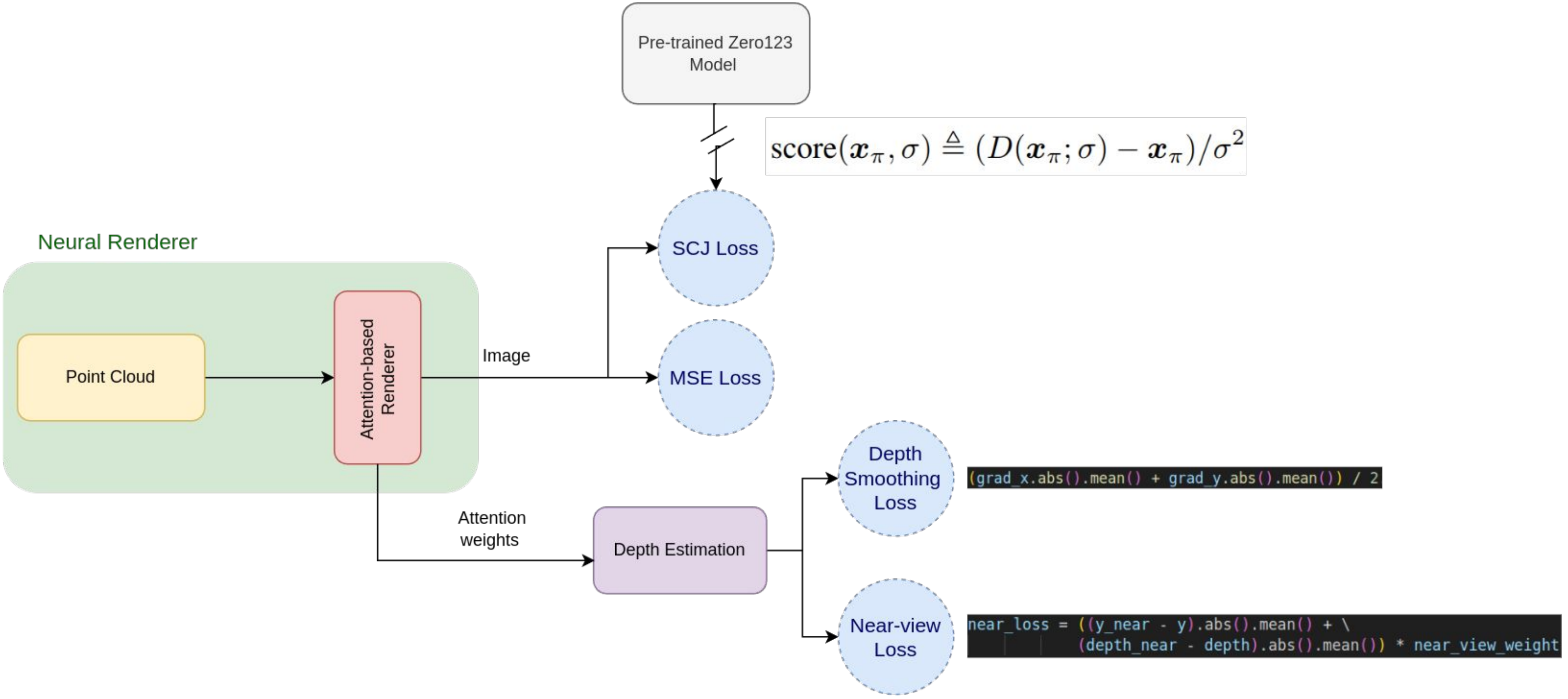
$$\underbrace{\nabla_{\boldsymbol{\theta}} \log \tilde{p}_\sigma(\boldsymbol{\theta})}_{\text{3D score}} = \mathbb{E}_\pi [\, \underbrace{\nabla_{\boldsymbol{x}_\pi} \log p_\sigma(\boldsymbol{x}_\pi)}_{\text{2D score; pretrained}} \cdot \underbrace{\boldsymbol{J}_\pi}_{\text{renderer Jacobian}} \,].$$

- A near-view consistency loss to regularize the change in appearance between nearby views
- Depth smoothing loss functions

Second Phase Challenges

- The first challenge was to set dependencies correctly
  - Some dependencies of PAPR and zero123 had conflict and some was missed
  - Opencv-python should not install in pip, but it should be loaded with the module command
  - …
- Zero123 used NeRF as the neural renderer which volumetric base while PAPR using point clouds as 3D assets
  - A number of functions should be rewritten or modified, which required a depth understanding of both zero123 and PAPR
  - GPU memory was an important concern to load both pre-trained zero123 model and PAPR which is based on attention mechanism
  - Changing hyper-parameters to meet the resources
    - Smaller patch size, lower number of points in point cloud
  - Slow debugging procedure owing to the computation intensity.
- PAPR does not directly generate depth estimation which is used for near-view consistency and depth smoothing losses
- After a good amount of time I come up with a code which properly run on cuda device with five loss functions and I trained it on chair benchmark for 10 hrs
  - SJC loss
  - Near-view loss
  - MSE loss
  - Input & target depth smoothing

Architecture & Loss Functions in a Nutshell

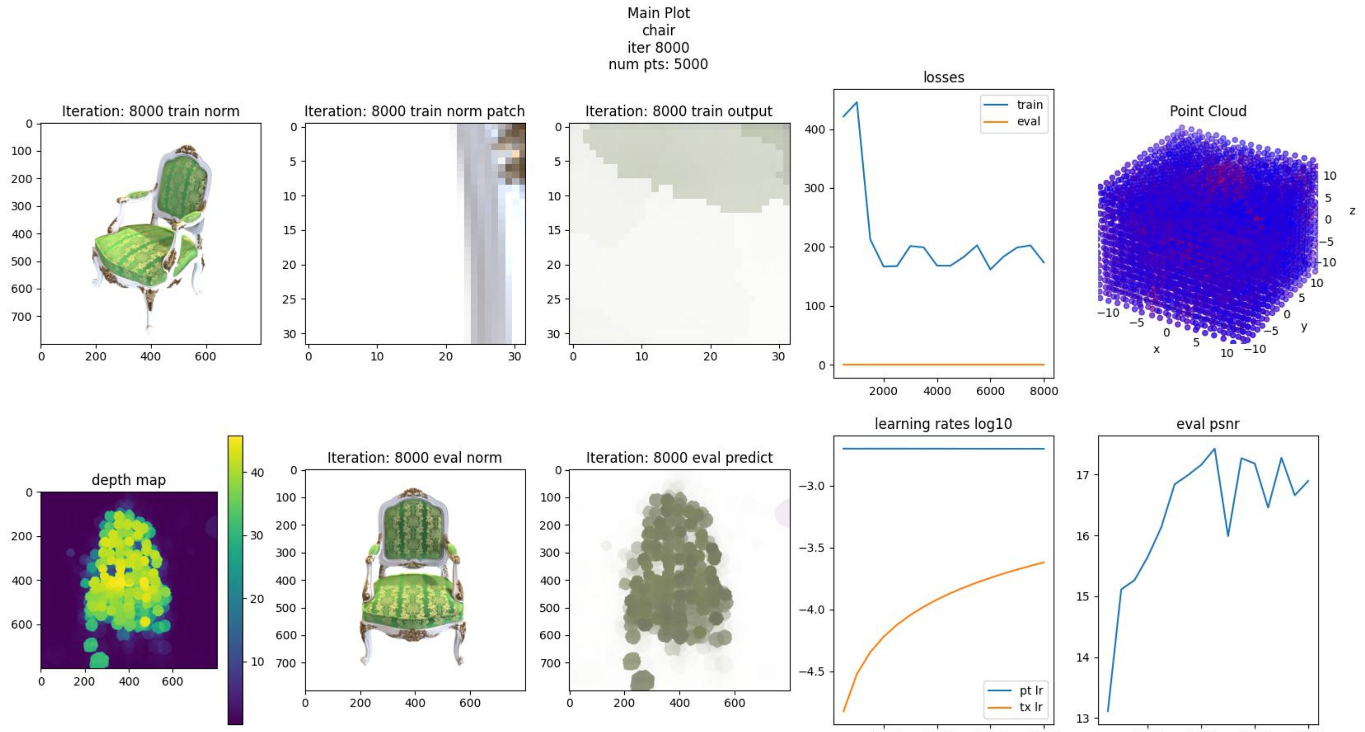


$$\text{score}(\boldsymbol{x}_\pi, \sigma) \triangleq (D(\boldsymbol{x}_\pi;\sigma) - \boldsymbol{x}_\pi)/\sigma^2$$

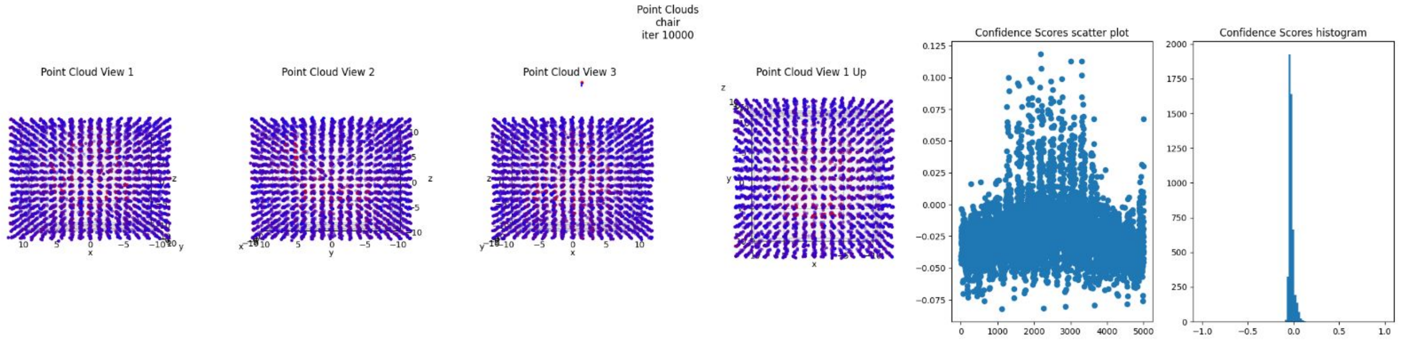
```
(grad_x.abs().mean() + grad_y.abs().mean()) / 2
```

```
near_loss = ((y_near - y).abs().mean() + \
        (depth_near - depth).abs().mean()) * near_view_weight
```

Second Phase Results

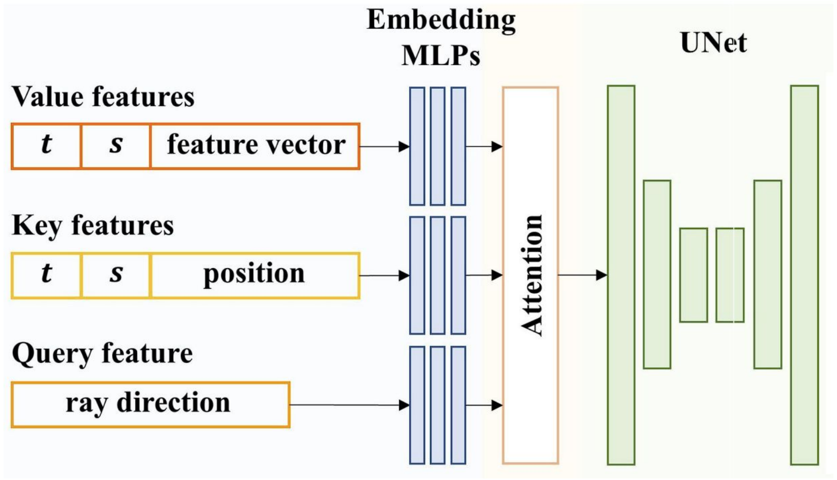


Second Phase Results cont.



Third Phase

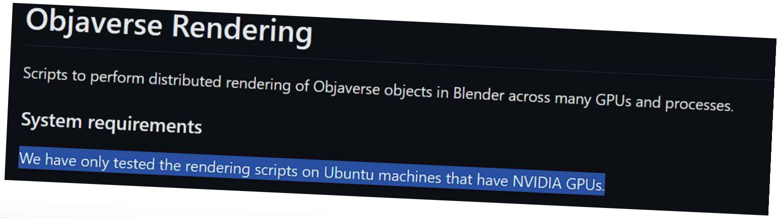
- Modify the PAPR framework to share weights across video frames and alter the point cloud.
  - For frames above base frame:
    - Freeze Unet
    - Reuse Point Cloud output of past frame
      - Features and Ray Directions should remain similar to past frame
    - Positions and MLP subject to change
- Refine the motion aspect by incorporating moving objects:
  - Set up Compute Canada for code execution
  - Download Model from Objaverse Source
  - Blender to render images and create json
  - Adjust PAPR framework to be compatible with new data and add temporal aspects
  - Run!



Challenges and Complications

- Setting up Compute Canada Env properly
  - StdEnv version complications (Resolved)
  - Globus complications (Resolved)
  - Package mismatch (Resolved)
    - Previously incompatible with Objaverse due whl errors
      - PyArrow missing
    - Still only compatible with outdated versions (< 1.0.0)
- Working with Objaverse (Workaround)
  - Under Documented API
  - Minimally developed render code
    - Incompatible with windows
- Working with Blender (Resolved)
  - Learning curve
  - Scripting environment setup

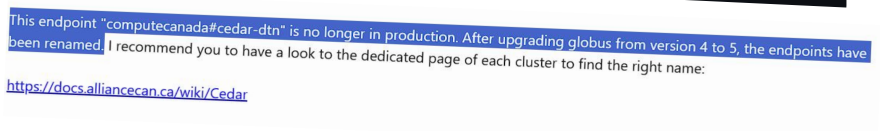
- Working with PAPR (Ongoing)
  - Complex codebase needing many adjustments of various sizes to work with video setup + custom data
  - Slow debugging process due to compute intensity
    - Compute Canada GPU queue


Objaverse Rendering

Scripts to perform distributed rendering of Objaverse objects in Blender across many GPUs and processes.

System requirements

We have only tested the rendering scripts on Ubuntu machines that have NVIDIA GPUs.

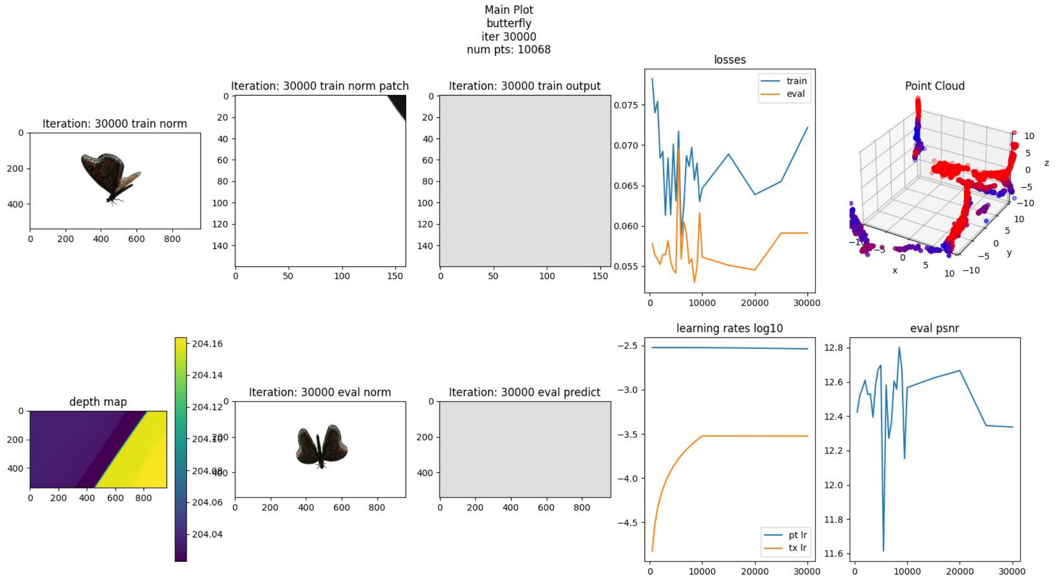

This endpoint "computecanada#cedar-dtn" is no longer in production. After upgrading globus from version 4 to 5, the endpoints have been renamed. I recommend you to have a look to the dedicated page of each cluster to find the right name:

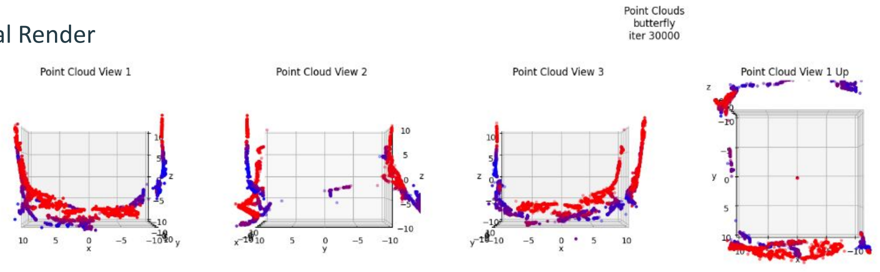
https://docs.alliancecan.ca/wiki/Cedar

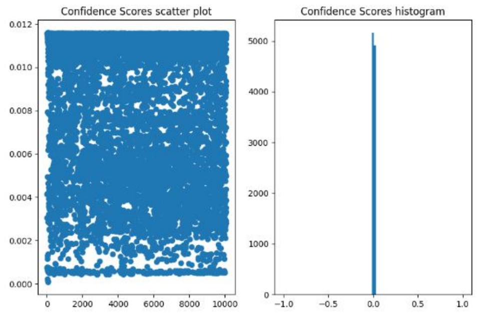
Results

Intermediate results from baseframe
Input: yml, json, images
Output: Model - pth file

Original Render







Steps to merge three phases

- We needed spend a good amount of time to get familiar with zer0123++ model because of inconsistency between stable diffusion versions of AnimateDiff and zero123
- After that, we should replace zero123 model in second phase with the combination of zero123++ and AnimateDiff
- Then we should replace the PAPR model in second phase with the modified version of it, in third phase, which is proper for a sequence of images
- Given a single image, get video from specific views through result of the first phase
- Use these sequence of images to train modified PAPR with these losses
  - MSE
  - SJC
  - Near-view smoothing
  - Input-depth smoothing
  - Target-depth smoothing