Rodin

Approach

- learn 3D avatar generation off multi-view renderings from 'Blender Synthetic Pipeline': Computer software that gives 2D images of 3D objects from known viewpoints

3D Training Data Representation

We need a way to represent 3D neural radiance fields that:

- Can input to Neural Networks
- Is memory efficient & fast

Vanilla NeRF approach would take too long to create a synthetic dataset

Tri-Plane Representation

- Each 3D point $p \in \mathbb{R}^3$ is mapped to 3 planar coordinates P_{uv}, P_{wu}, P_{vw} for planes uv, wv, vw representing each axis.
- The density & colour of each point is encoded on a planar level as embeddings y_{uv}, y_{wu}, y_{vw}
- The overall embedding is computed as $y_p = y_{uv} P_{uv} + y_{wu} P_{wu} + y_{vw} P_{vw}$ which represents the colour & density of point p from different viewing angles as a number vector
- A lightweight MLP decoder then can extract the colour & density from each y_p given p and viewing direction d
- We also apply a positional encoding here to shift focus to high frequency details.

3D Diffusion Model

- Learns tri-plane features (tri-plane encoding of the colour & density) $P(y)$ where $y = (y_{uv}, y_{wu}, y_{vw})$ based off diffusion process

- Initial Approach: we have 3 planes $\in \mathbb{R}^{W \times H \times c}$, concatenate the 3 planes to make 1 image $\in \mathbb{R}^{W \times H \times 3c}$ & use current SOTA 2D diffusion models

  Problem: we lose spatial properties since each plane is no longer seperate but on top of each other

3D Aware Concatenation

- For each point y, for the plane y_{uv}, add in some information that tells it about the other planes at y: y_{vw}, y_{wu}. Now do for the other planes.
  - by taking the line intersection of the 2 planes, embedding it & then concatenating this info into y_{uv}.
- Allows colour & density at y_{uv} to depend on y_{wu} & y_{vw}

Latent Conditioning

- Use CLIP to represent the front view of the image as a latent vector (which CLIP can understand with text)
- Now as the diffusion model constructs images, it checks with CLIP that it is creating the correct coherent features. CLIP can now also manipulate the generative model with text input.

AdaGN framework

Model Upsampler

- After diffusion model outputs a 64×64 image, we train a network to upsample to 256×256 images to enhance quality by feeding the network diffusion images & ground truth.
- Also train a convolution refiner to find high-frequency details that NeRF misses taking the images to 1024×1024.

all done using tri-plane coordinate system

What can the model do?

Avatar Portrait

- Input: Single input image
  Output: 3D renderable avatar

CLIP just feeds the image's own latent vector to diffusion network

Text → Avatar

- Input: text description
  Output: 3D renderable avatar

Customisable Avatar

CLIP takes current latent vector z & adds relevant features δ to output $z+\delta$

- Input: input image / current avatar & text description of changes (ie. "add glasses")
- Output: 3D renderable avatar