

基于 Diffusion 的三维模型生成

摘要

本文首先讨论了传统三维重建以及 NeRF 的局限性，然后就如何利用 Diffusion 2D 模型监督一个 NeRF 提出了解决方案，事实表明所提出的概率密度蒸馏的方式能够有效地生成所需的三维模型。为了改善生成三维模型的质量和速度，本文又在原始 NeRF 网络上做了一些改进，将生成场景限制在一个球体内，同时使用一个环境贴图作为背景。此外，为了改善模型生成带有头发人物肖像效果不佳的问题，参考了 HAAR 的 Pipeline，引入了一个中间层的表示，最终生成了高质量的头发模型。

关键词：扩散模型；三维重建；头发重建

1 引言

这几年，以 Stable Diffusion 和 Midjourney 为代表的 AI 绘图技术取得了突破性的进展，人们可以用简单的文字生成魔法般的 2D 图片。但是人们似乎还没有一个很好的办法从文本直接生成三维的物体，其中决定性的原因就在于数据上的鸿沟。著名的 Stable Diffusion 1.5 是在 LAION-5B 上展开的训练，而前些年，最大的 3D 数据集 Objaverse 甚至只有 700K 的大小，且 Objaverse 在三维界已经算是相当大的一个数据集了，但这仍然和 2D 数据有着几个数量级的差距。

数据量上的差异决定了人们很难直接训练出泛化性良好的三维生成模型。数据量上的差异决定了人们很难直接训练出泛化性良好的三维生成模型，因此有没有一种方式能够利用 Stable Diffusion 和 Midjourney 这样的 AI 绘图模型的知识作为先验，来生成一个个的三维物体就显得尤为重要。

2 相关工作

2.1 NeRF

2020 年开始，NeRF 的快速发展使得我们可以从一系列真实世界的照片中重建出一个真实的世界。现如今，在消费级 GPU 上，我们已经能够以超过 100FPS 的速度，渲染出难辨真假的视频。然而，NeRF 达到如此高的渲染精度是有一定条件的，那就是给定的数据需要较为精确。不仅你要明确地告诉机器拍摄时候相机的姿势和相机的焦距，同时被拍摄的物体也不能出现太大的形变或者位移，物体本身也不能被更换成其他物体。那现在假设我们要利用 Diffusion Model 生成的图片直接来监督 NeRF，会产生什么问题呢？

不巧的是，上面提到的三个条件都很难被满足：

1. 以 Stable Diffusion 为例，输出的结果仅仅能通过粗略的语言描述控制，如“Back View”, “Front View”, “From Aside”. 没有办法准确的用相机的参数来控制结果的视角，更不用说焦距的指定了。
2. 相应的，物体的形状也没有办法很好的保持，即使运用了 LoRA 或者 DreamBooth 之类的技术来生成特定的角色，在服饰细节上也没有办法很好的保持
3. 一些更抽象和泛化的描述，比如“穿着黑色夹克的男孩”，多次生成的过程中出来的人甚至都不是同一个。在这种情况下重建几乎是不可能的。

因此，直接利用 diffusion 模型生成的 2D 图片来做几何重建是非常困难的一件事。但 diffusion 里面又拥有着非常丰富的 2D 信息和语言知识，如果不能加以利用的话实在是太过可惜。如果不能直接地利用它生成地图片来合成 3D 内容的话，该有没有一种方式可以间接地利用 diffusion 模型地内在知识呢？这就要引出 DreamFusion 的解决思路了。

2.2 DreamFusion

DreamFusion [9] 提出了如何间接地利用 diffusion 模型监督一个 NeRF 的方法，那就是概率密度蒸馏函数。概率密度蒸馏函数也是 DreamFusion 的一大创新点，它通过直接去掉 UNet 项的梯度，不仅解决了梯度消失问题，很好地提高了生成模型的品质。同时，它也极大了提高了生产三维模型的速度。概率密度蒸馏函数的泛用性也很强，任何一个能够生成图片的生成器都可以利用概率密度蒸馏函数来优化。

尽管当前已经有很多 Text-to-3D 的模型效果远超 Dreamfusion [7] [16] [5] [15] [8] [3] [14] [12] [4] [10] [6] [17] [2]，但是作为该领域的开山之作，仍具有重大意义。

3 本文方法

3.1 本文方法概述

本文的方法管线图如图1所示。当我们想用 Diffusion 去噪的过程来监督一个 NeRF 时，最简单的方式就是模仿 Diffusion 的训练过程，去给 NeRF 渲染出来的图片也加上类似的噪声。这个时候假如 NeRF 渲染出来的图片接近真实的图片，那么我们用 Diffusion 模型预测的噪声就应该和加上的噪声接近。此时，我们就可以照搬 Diffusion 的损失函数来度量噪声的接近程度：

$$L_{NeRF} = E[w(t) \|\text{UNet}(\alpha_t x_{\text{render}} + \sigma_t \varepsilon | t) - \varepsilon\|^2]$$

概括来说，就是左边的 NeRF 先渲染出一张孔雀的图片，加噪后利用 diffusion 模型预测出相应噪声，再和真正的噪声作对比。然后用对比的结果反向传播优化 NeRF 的渲染结果，从而生成逼真的三维模型。

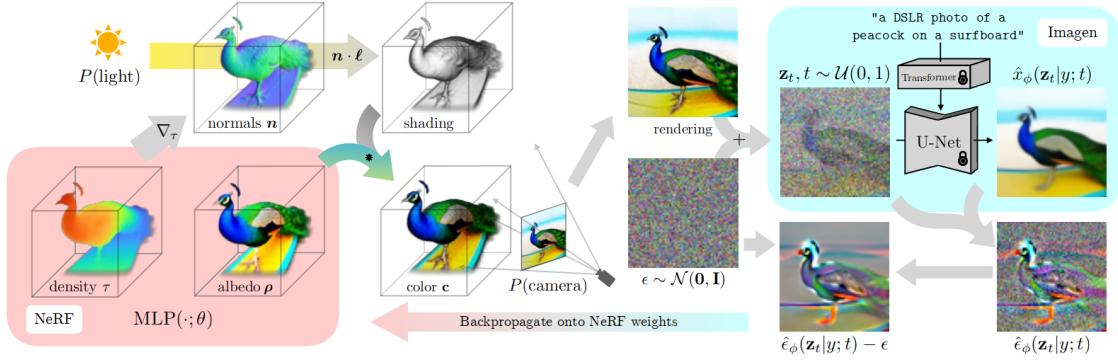


图 1. 方法示意图

3.2 损失函数定义

然而，观察上述的损失函数

$$L_{NeRF} = E[w(t) \|UNet(\alpha_t x_{render} + \sigma_t \varepsilon | t) - \varepsilon\|^2]$$

我们会发现，计算这个损失函数会反向传播计算 UNet 网络的梯度，如果优化的每一步都要计算 UNet 的梯度，那么将会相当慢。DreamFusion 最重要的地方就是它提出了 SDS Loss，规避了 UNet 梯度的计算。简单的推导如下，利用链式法则将该梯度拆分

$$\frac{dL_{NeRF}}{dz} \cdot \frac{dz}{d\theta} = 2\omega(t) \|UNet(z|t) - \varepsilon\| \cdot \frac{dUNet(z|t)}{dz} \cdot \alpha_t I \cdot \frac{dx_{render}}{d\theta}$$

为了简洁一些，我们合并含 \$t\$ 的项和常数项，称为 \$\omega'(t) = 2\omega(t)\alpha_t I\$，这样我们可以得到

$$grad_{SDS} = \omega'(t) \|UNet(z|t) - \varepsilon\| \cdot \frac{dUNet(z|t)}{dz} \frac{dx_{render}}{d\theta}$$

其中第一项是 diffusion 的噪声残差，第二项是 Unet 部分对应的梯度，第三部分就是 NeRF 对应的梯度。这个时候，Google 的研究人员大手一挥，你不是 Unet 部分对应的梯度不好算嘛，直接去掉就好了。于是就有了最后的 SDS 梯度：

$$grad_{SDS} = \omega'(t) \|UNet(z|t) - \varepsilon\| \cdot \frac{dx_{render}}{d\theta}$$

3.3 SDS 损失函数理论解释

$$\nabla_\theta \mathcal{L}_{SDS}(\phi, \mathbf{x} = g(\theta)) \triangleq \mathbb{E}_{t,\epsilon} \left[w(t) (\hat{\epsilon}_\phi(\mathbf{z}_t | y, t) - \epsilon) \frac{\partial \mathbf{x}}{\partial \theta} \right]$$

使用上述损失函数，效果变好了，下面我们就讨论这个损失函数的数学原理。

$$\begin{aligned} \nabla_\theta \mathcal{L}_{SDS} &= \mathbb{E}_{t,\mathbf{z}_t|m} m_{\mathbf{x}} \left[w(t) \frac{\sigma_t}{\alpha_t} \nabla_\theta \text{KL}(q(\mathbf{z}_t | \mathbf{x} = g(\theta)) \| p_\phi(\mathbf{z}_t | y)) \right] \\ &= \mathbb{E}_{t,\epsilon} \left[w(t) (\hat{\epsilon}(\mathbf{z}_t | y) - \epsilon) \frac{\partial \mathbf{x}}{\partial \theta} \right] \end{aligned}$$

因为 p_ϕ 和 q 都是密度分布，且 p_ϕ 是已知的，所以有点类似知识蒸馏那边，一个已知网络训练一个未知网络的过程，故称之为概率密度蒸馏。

现在来算一下：

$$\text{KL}(q(\mathbf{z}_t | \mathbf{x} = g(\theta)) \| p_\phi(\mathbf{z}_t | y)) = \mathbb{E}_\epsilon [\log q(\mathbf{z}_t | \mathbf{x} = g(\theta)) - \log p_\phi(\mathbf{z}_t | y)]$$

$$\nabla_\theta \text{KL}(q(\mathbf{z}_t | \mathbf{x} = g(\theta)) \| p_\phi(\mathbf{z}_t | y)) = \mathbb{E}_\epsilon [\underbrace{\nabla_\theta \log q(\mathbf{z}_t | \mathbf{x} = g(\theta))}_{(A)} - \underbrace{\nabla_\theta \log p_\phi(\mathbf{z}_t | y)}_{(B)}]$$

先来算 B 项目：

$$\begin{aligned} & \nabla_\theta \log p_\phi(z_t | y) \\ &= \nabla_{z_t} \log p_\phi(z_t | y) \frac{\partial z_t}{\partial \theta} \\ &= s_\phi(z_t | y) \alpha_t \frac{\partial x}{\partial \theta} \\ &= -\frac{\alpha_t}{\sigma_t} \epsilon_\phi(z_t | y) \frac{\partial z_t}{\partial \theta} \end{aligned}$$

再来看 A 项目：

$$\frac{\partial \log q(z_t | x)}{\partial x} \frac{\partial x}{\partial \theta} + \frac{\partial \log q(z_t | x)}{\partial z_t} \frac{\partial z_t}{\partial x} \frac{\partial x}{\partial \theta}$$

我们将左边部分称为 C 项，右边部分称为 D 项。考虑当

$$q(z_t | x) = p_\phi(z_t | y)$$

A 项会变为：

$$\begin{aligned} & \frac{\partial \log p_\phi(z_t | y)}{\partial x} \frac{\partial x}{\partial \theta} + \frac{\partial \log p_\phi(z_t | y)}{\partial z_t} \frac{\partial z_t}{\partial x} \frac{\partial x}{\partial \theta} \\ &= \frac{\partial \log p_\phi(z_t | y)}{\partial x} \frac{\partial x}{\partial \theta} + \nabla_{z_t} \text{situ } gp_\phi(z_t | y) \frac{\partial z_t}{\partial \theta} \end{aligned}$$

所以当两个分布一样时，D-B 一定为 0，但是 A-B=(C+D)-B 就不一定了，为了减少结果的方差，进行无偏估计，论文里舍弃掉了 A 中的 C 项，只保留了 D 项。所以 A 项为：

$$\begin{aligned} & \nabla_\theta \log q(z_t | x = g(\theta)) \\ &= \frac{\partial \log q(z_t | x)}{\partial z_t} \frac{\partial z_t}{\partial x} \frac{\partial x}{\partial \theta} \end{aligned}$$

由于：

$$\begin{aligned} z_t &\sim N(\alpha_t x, \sigma_t^2 I) \\ z_t &= \frac{1}{\sqrt{2\pi}\sigma_t} e^{-\frac{(z_t - \alpha_t x)^2}{2\sigma_t^2}} \\ \log p(z_t | x) &= -\frac{(z_t - \alpha_t x)^2}{2\sigma_t^2} - \log \sqrt{2\pi} \end{aligned}$$

因此：

$$\begin{aligned} & \nabla_\theta \log q(z_t | x = g(\theta)) \\ &= -\frac{1}{2\sigma_t^2} 2(z_t - \alpha_t x)(-\alpha_t) \frac{\partial x}{\partial \theta} \\ &= -\frac{1}{\sigma_t^2} \sigma_t \epsilon(-\alpha_t) \frac{\partial x}{\partial \theta} \\ &= \frac{\alpha_t}{\sigma_t} \epsilon \frac{\partial x}{\partial \theta} \end{aligned}$$

所以，有：

$$\begin{aligned}\nabla_{\theta} L_{SDS} &= E_{t,\epsilon} [w(t) \nabla_{\theta} KL(q(z_t | x = g(\theta)) \| p_{\phi}(z_t | y))] \\ &= E_{t,\epsilon} \left[w(t) \left(-\frac{\alpha_t}{\sigma_t} \epsilon \frac{\partial x}{\partial \theta} + \frac{\alpha_t}{\sigma_t} \epsilon_{\phi}(z_t | y) \frac{\partial z_t}{\partial \theta} \right) \right] \\ &= E_{t,\epsilon} \left[\frac{\alpha_t}{\sigma_t} w(t) (\epsilon_{\phi}(z_t | y) - \epsilon) \frac{\partial x}{\partial \theta} \right]\end{aligned}$$

所以，可以证明我们之前的损失函数的梯度，正是这两个分布的 KL 散度的梯度。

4 复现细节

4.1 与已有开源代码对比

复现时参考了 stable-dreamfusion 的开源代码 [1]，但是此代码生成模型速度较慢，同时精度有限。此外尤其是对于带有头发的人物模型的生成效果非常差，如图6所示。可以看到想要 DreamFusion 生成一张带头发的女性角色的模型，生成效果是十分之差的。

为了解决生成头发模型效果差的问题，我们引入了图3中的 HAAR 管线图 [13]。这个管线引入了一种名为 Neural Strands [11] 的发表示中间层，这个中间层将头发顶点数据编码为 [1,64,32,32] 的 tensor，然后利用 Diffusion 模型生成头发数据，再用解码器进行解码，生成最后的发丝。

4.2 实验环境搭建

实验环境为 Ubuntu torch 版本 1.12 CUDA 版本 11.6 显卡为 2080ti。实验时代码运行图，如图2所示。

```

按文件名过滤
+ /autodl-tmp/stable-dreamfusion /          父级目录
名称   修改时间
assets  18小时前
config  18小时前
data    18小时前
docker  18小时前
evaluation 18小时前
freqencoder 2小时前
gridencoder 4小时前
guidance  2小时前
ldm     18小时前
nerf    4小时前
pretrained 4小时前
raymarching 4小时前
scripts  18小时前
shencoder 18小时前
taichi_mod... 18小时前
tets    18小时前
trial   1小时前
activation.py 18小时前
dpt.py   18小时前
encoding.py 18小时前
LICENSE  18小时前
main.py   18小时前
meshutils.py 18小时前
optimizer.py 18小时前
preprocess... 18小时前
readme.md  18小时前
requiremen... 3小时前

终端 3
[ INFO ] nro=1, lambda_rgb=1000
[ INFO ] lambda_mask=500, lambda_normal=0, lambda_depth=10, lambda_2d_normal_smooth=0
[ INFO ] lambda_3d_normal_smooth=0,
[ INFO ] save_guidance=False, save_guidance_interval=10, gui=False, W=800, H=800, rad
ius=5, Fov=20, light_theta=60,
light_phi=0, max_spp=1, zeroi23_config= ./pretrained/zeroi23/sd-objaverse-fi
netune_r_concat=256 yaml',
zeroi23_ckpt='pretrained/zeroi23/zeroi23-xl.ckpt', zeroi23_grad_scale='angle
', dataset_size_train=100,
dataset_size_val=8, dataset_size_test=100, exp_start_iter=0, exp_end_iter=
10000, images=None,
ref_radius=[], ref_polars=[], ref_azimuths=[], zeroi23_ws=[], default_zeroi23
_w=1]
[INFO] Trainer: df | 2024-12-04_18:00:59 | cuda | fp16 | trial
[INFO] #parameters: 12284151
[INFO] Loading latest checkpoint ...
[INFO] Latest checkpoint is trial/checkpoints/df_ep0100.pth
[INFO] loaded model.
[INFO] load at epoch 100, global step 10000
[WARN] Failed to load optimizer.
[INFO] loaded scheduler.
[INFO] loaded scales.
[INFO] Start Test, save results to trial/results
100% 100/100 [00:04<00:00, 30.951/s]*** Finished Test.
100% 100/100 [00:06<00:00, 16.651/s]
root@autodl-container-aa414bbc9b-1f14c75e:~/autodl-tmp/stable-dreamfusion# l
ls
[LICENSE config encoding.py guidance nerf rayma
xching shencoder _pycache__ data evaluation ldm optimiz...
e.md taichi_modules activation.py docker freqencoder main.py preprocess_image.py requi
rements.txt tets assets dpt.py gridencoder meshutils.py pretrained scrip
ts trial
root@autodl-container-aa414bbc9b-1f14c75e:~/autodl-tmp/stable-dreamfusion# c
d trial/
root@autodl-container-aa414bbc9b-1f14c75e:~/autodl-tmp/stable-dreamfusion/tr
ial# ls
checkpoints log_df.txt results run validation
root@autodl-container-aa414bbc9b-1f14c75e:~/autodl-tmp/stable-dreamfusion/tr
ial# cd results/
root@autodl-container-aa414bbc9b-1f14c75e:~/autodl-tmp/stable-dreamfusion/tr
ial# ls
df_ep0100_depth.mp4 df_ep0100_rgb.mp4
root@autodl-container-aa414bbc9b-1f14c75e:~/autodl-tmp/stable-dreamfusion/tr
ial# /results# []

```

图 2. 实验环境

4.3 界面分析与使用说明

main.py 负责整个实验的入口程序，比如生成一个汉堡是如下命令：

python main.py -text “a hamburger” -workspace trial -O 此命令给出了 diffusion 模型所要用的提示词，以及输出的工作区。

利用 dmtet 加强已有的汉堡的代码为：

python main.py -O -text “a hamburger” -workspace trial_dmtet -dmtet -iters 5000 -init_with trial/checkpoints/df.pth 此命令读取了已有的汉堡的工作区，然后迭代 5000 次来优化整个模型。

4.4 创新点

本文的创新点在于解决了 stable-dreamfusion 对于人物头发模型生成效果极差的问题。

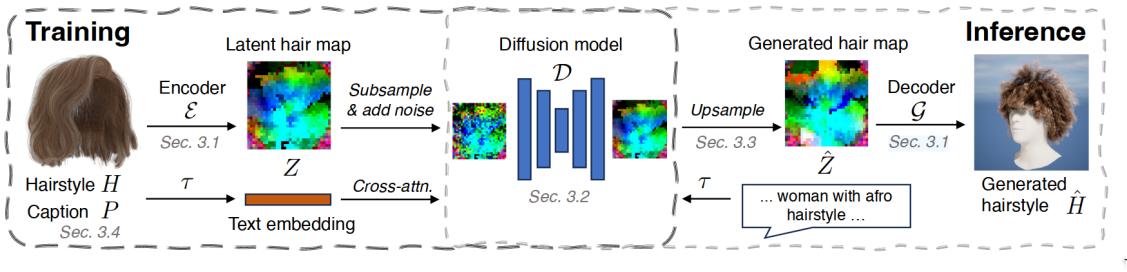


图 3. HAAR 管线图

可以看到，这个管线体现了，一个思想：“在计算机中，任何问题都可以引入中间层来解决”。这其实与 Dreamfusion 的思想是一致的，它们都是利用间接的方法生成我们想要的 3D 模型。

在 HAAR 中，我们为了生成逼真的发丝模型，我们不需要让 Diffusion Model 直接生成对应的图片，而是定义一种中间层的表示，训练 Diffusion Model 生成这种中间层的表示，再由中间层的表示生成发丝模型。这种表示很好地解决了传统的 SDS Loss 的只能捕获 3D 模型的表面结构和无法表示头发内部结构的问题，同时取得了更快的速度。传统的 SDS Loss 往往需要数个小时，而这种表示只需要一分钟不到。

5 实验结果分析

图4为仅使用 DreamFusion 生成的汉堡模型，可以看到这个汉堡模型的精度不高细节十分模糊。

图5为使用 dmtet 强化细节后的汉堡模型，可以看到这个模型的效果已经十分逼真了，汉堡肉和洋葱片以及芝士都清晰可见。

图6为使用 DreamFusion 生成的长发女士模型，可以看到这个生成效果十分差，尤其是头发部分模糊不可见。

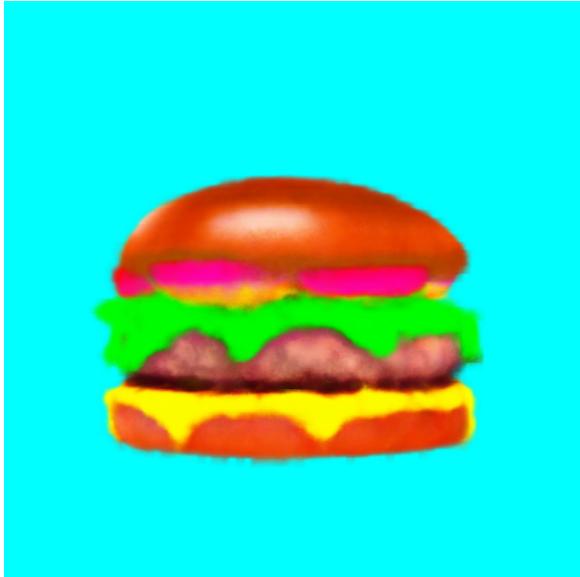


图 4. 汉堡

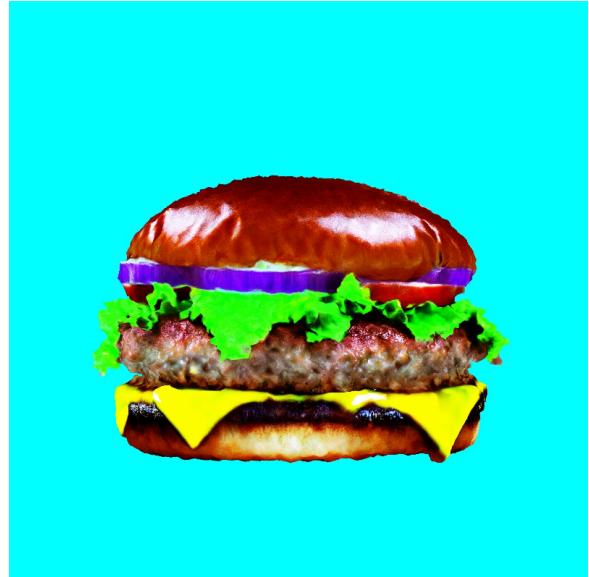


图 5. 汉堡 (增强)

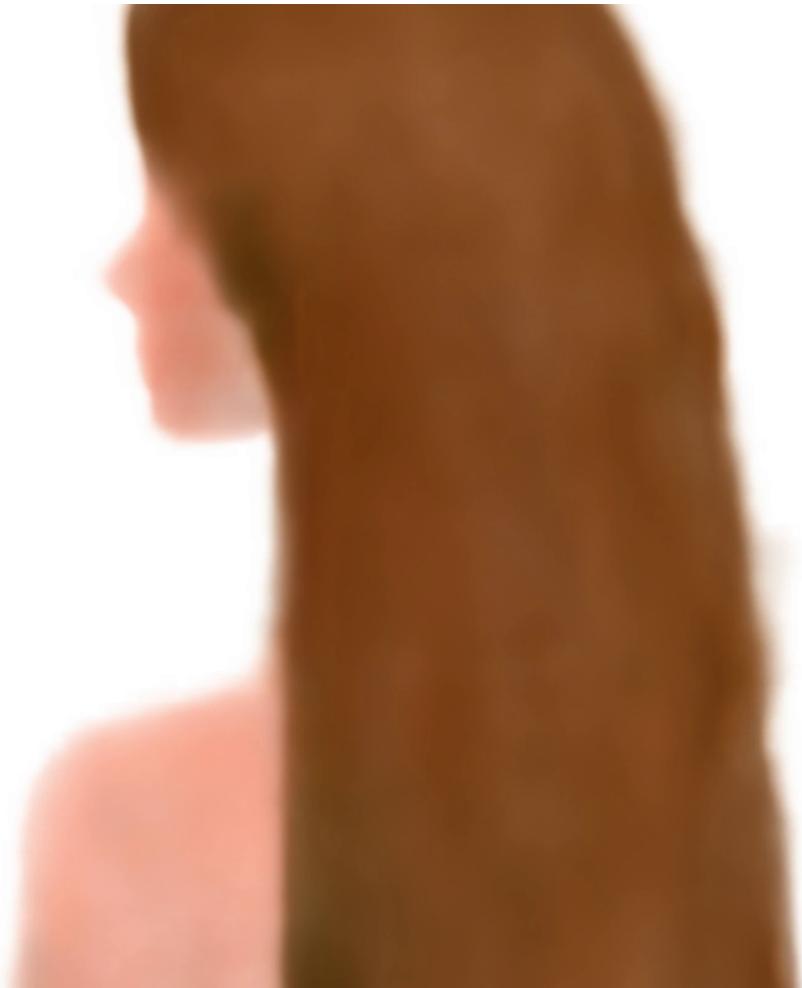


图 6. 长发女生

图7和图8为使用 HAAR 生成的头发模型，可以看到这个模型的头发效果十分好，头发都是一根根的，细节丰富，同时利于物理模拟。



图 7. HAAR 结果 1



图 8. HAAR 结果 2

6 总结与展望

- 首先我们知道，Diffusion 模型直接生成的结果难以控制，缺乏空间上的一致性，这让 NeRF 几乎不可能直接从 Diffusion 生成的图片中直接进行优化。
- 因此，DreamFusion 巧妙地利用了 Diffusion 模型的生成过程来做优化，而非利用生成结果来做优化。为了达成这一目标，DreamFusion 利用 Unet 去噪的能力，来间接地判断了 NeRF 生成地结果是否符合 Diffusion 模型的先验知识。
- 然而，直接从损失函数计算梯度要囊括复杂的 Unet 梯度计算。为了简化这一过程，Dream-

Fusion 利用链式法则将梯度计算加以分解。最终，通过经验上的结果略掉了 Unet 的梯度部分，这也是 SDS Loss 最核心的一步。

- SDS Loss 不是万能的，针对特殊的问题，我们要善于分析问题的本质，利用计算机科学中的思想探索最佳的解决方案。

参考文献

- [1] ashawkey/stable-dreamfusion: Text-to-3D & Image-to-3D & Mesh Exportation with NeRF + Diffusion.
- [2] Yukang Cao, Yan-Pei Cao, Kai Han, Ying Shan, and Kwan-Yee K. Wong. DreamAvatar: Text-and-Shape Guided 3D Human Avatar Generation via Diffusion Models, November 2023. arXiv:2304.00916 [cs].
- [3] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3D: Disentangling Geometry and Appearance for High-quality Text-to-3D Content Creation, September 2023. arXiv:2303.13873 [cs].
- [4] Ayaan Haque, Matthew Tancik, Alexei A. Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D Scenes with Instructions, June 2023. arXiv:2303.12789 [cs].
- [5] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-Resolution Text-to-3D Content Creation, March 2023. arXiv:2211.10440 [cs].
- [6] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot One Image to 3D Object, March 2023. arXiv:2303.11328 [cs].
- [7] Artem Lukoianov, Haitz Sáez de Ocáriz Borde, Kristjan Greenewald, Vitor Campagnolo Guizilini, Timur Bagautdinov, Vincent Sitzmann, and Justin Solomon. Score Distillation via Reparametrized DDIM, October 2024. arXiv:2405.15891 [cs] TLDR: This work shows that the image guidance used in Score Distillation can be understood as the velocity field of a 2D denoising generative process, up to the choice of a noise term, which makes SDS's generative process for 2D images almost identical to DDIM.
- [8] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-NeRF for Shape-Guided Generation of 3D Shapes and Textures, November 2022. arXiv:2211.07600 [cs].
- [9] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion, September 2022. arXiv:2209.14988 TLDR: This work introduces a loss based on probability density distillation that enables the use of a 2D diffusion model as

a prior for optimization of a parametric image generator in a DeepDream-like procedure, demonstrating the effectiveness of pretrained image diffusion models as priors.

- [10] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. Magic123: One Image to High-Quality 3D Object Generation Using Both 2D and 3D Diffusion Priors, July 2023. arXiv:2306.17843 [cs] TLDR: A two-stage coarse-to-fine approach for high-quality, textured 3D meshes generation from a single unposed image in the wild using both 2D and 3D priors, as validated through extensive experiments on synthetic benchmarks and diverse real-world images.
- [11] Radu Alexandru Rosu, Shunsuke Saito, Ziyan Wang, Chenglei Wu, Sven Behnke, and Giljoo Nam. Neural Strands: Learning Hair Geometry and Appearance from Multi-View Images, July 2022. arXiv:2207.14067 TLDR: This work proposes a novel hair representation based on a neural scalp texture that encodes the geometry and appearance of individual strands at each texel location, and introduces a novel neural rendering framework based on rasterization of the learned hair strands.
- [12] Ruizhi Shao, Jingxiang Sun, Cheng Peng, Zerong Zheng, Boyao Zhou, Hongwen Zhang, and Yebin Liu. Control4D: Efficient 4D Portrait Editing with Text, November 2023. arXiv:2305.20082 [cs] TLDR: Control4D, a novel approach for high-fidelity and temporally consistent 4D portrait editing, is proposed, built upon an efficient 4D representation with a 2D diffusion-based editor and learns a 4D GAN from it and avoids the inconsistent supervision signals.
- [13] Vanessa Sklyarova, Egor Zakharov, Otmar Hilliges, Michael J. Black, and Justus Thies. Text-Conditioned Generative Model of 3D Strand-Based Human Hairstyles. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4703–4712, Seattle, WA, USA, June 2024. IEEE. TLDR: HAAR is a first text-guided generative method that uses 3D hair strands as an underlying representation that produces 3D hairstyles that could be used as production-level assets in modern computer graphics engines.
- [14] Christina Tsalicoglou, Fabian Manhardt, Alessio Tonioni, Michael Niemeyer, and Federico Tombari. TextMesh: Generation of Realistic 3D Meshes From Text Prompts, April 2023. arXiv:2304.12439 [cs].
- [15] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score Jacobian Chaining: Lifting Pretrained 2D Diffusion Models for 3D Generation, December 2022. arXiv:2212.00774 [cs].
- [16] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation, November 2023. arXiv:2305.16213 [cs] TLDR: This work proposes to model

the 3D parameter as a random variable instead of a constant as in SDS and presents variational score distillation (VSD), a principled particle-based variational framework to explain and address the aforementioned issues in text-to-3D generation.

- [17] Jiale Xu, Xintao Wang, Weihao Cheng, Yan-Pei Cao, Ying Shan, Xiaohu Qie, and Shenghua Gao. Dream3D: Zero-Shot Text-to-3D Synthesis Using 3D Shape Prior and Text-to-Image Diffusion Models, April 2023. arXiv:2212.14704 [cs].