从单目视频重建个性化语义面部 NeRF 模型

摘要

本研究改进了使用神经辐射场（NeRF）的方法，以实现个性化的人头表示。原方法中，每个基础都具有特定的语义含义，通过低维码生成人头图像。本研究引入了多级体素场与潜在空间中的表情系数相结合的表示，有效地学习不同尺度的头部细节。通过线性混合设计，调节局部特征以适应 MLP 的输入分布，进而产生更逼真的面部细节。研究还通过在表面附近集中采样来提高渲染效率，NeRF 基础的构建时间不到 20 分钟，训练模型具有交互式渲染速度，并能生成逼真的人头图像。该表示允许进行语义编辑，如调整相机参数和改变表情，同时保持其他属性不变。工作流程包括使用现存基于网格的面部 blendshape 框架追踪输入视频中的面部，得到表情系数和头部姿势参数。通过将人头姿势参数作为相机参数，将每帧的基础几何对齐到相同的空间位置。随后，从输入视频中随机提取 N 帧训练模型，并通过分割方法生成所选帧的头部掩模。训练模型的可学习变量包括隐式函数的网络参数和表示表情基底的多级哈希表。损失项包含光度损失、掩模损失和知觉损失。为了最大限度地减少总损失，提出了精心设计的训练策略。考虑到特定表情的密度场无法覆盖所有情况，文章设计了一种表情感知的密度网格更新策略，用于进一步加速体渲染中的 ray marching，并使优化集中在可能占据头部区域。

关键词： NeRF；个性化人头表示

1 引言

该论文提出的方法使用神经辐射场（NeRF）来进行个性化的人头表示。模型的每个基础都具有特定的语义含义，通过低维码可以轻松生成所需的人头图像。该表示将多级体素场与潜在空间中的表情系数相结合，可以有效地学习不同尺度的头部细节。线性混合设计调节局部特征以适应 MLP 的输入分布，从而产生更逼真的面部细节。该方法还通过在表面附近集中采样来提高渲染效率。NeRF 基础的构建时间不到 20 分钟，训练模型具有交互式渲染速度，并能生成逼真的人头图像。该表示允许语义编辑，如调整相机参数和改变表情，同时保持其他属性不变。

大概的工作如下：首先，它使用现存基于网格的面部 blendshape 框架 [1]，来追踪输入视频中的面部，得到每帧的表情系数和头部姿势参数。框架将人头姿势参数作为相应帧的外部相机参数，隐式地将每帧的基础几何对齐到相同的空间位置。最后框架从输入视频中随机提取 N 帧来训练模型，并通过现有的分割方法生成所选帧的头部掩模。

训练模型时，其可学习变量包括隐式函数的网络参数和表示表情基底的多级哈希表 [5]。训练模型的损失项包含以下三项：光度损失，掩模损失，知觉损失。为了最大限度地减少总

损失，文章提出了精心设计的训练策略。由于特定表情的密度场无法覆盖所有表情情况，文章设计了一种表情感知的密度网格更新策略，用于进一步加速体渲染中的 ray marching，并使优化集中在头部可能占据的区域。

2 相关工作

2.1 参数化头部模型

在人类头部形状空间可以很好地分解为身份、表情和外观的假设下，Blanz 和 Vetter 提出了 3DMM [6]，将 3D 头部形状嵌入到几个低维 PCA 空间中。以下大量工作对基于网格的参数化头部模型进行了进一步的研究。为了提高其表示能力，一些工作将其扩展到多线性模型，非线性模型以及具有校正混合形状的铰接模型以提高其建模能力。基于网格的方法和基于深度学习的方法都已广泛应用于许多相关应用中。然而，基于网格的参数模型由于其表示能力有限，通常无法表示个性化的面部细节。同时，现有的基于网格的参数模型无法表示非面部部分，尤其是头发。一些作品使用变形传递来处理这个问题或神经网络获得用户特定的混合形状基础。为了突破基于显式网格的数字人体表示的有限表示能力，许多工作采用隐式表示来提高模型容量和视觉质量。3DMM 是第一个基于神经隐式函数的全头 3D 可变形模型。HeadNeRF [4] 提出了一种基于神经辐射场的通用头部参数模型。尽管基于神经隐函数的表示已经表现出强大的表示能力，但通用模型通常仍然缺乏个性化的面部细节。NerFACE [2] 提出了一种基于 NeRF 的个性化人体头部模型。然而，他们的方法需要很长时间对每个主题进行训练和推理。IM 头像 [7] 提出了隐式 LBS 模型。请注意，该文的方法和 IMAvatar 都有隐式的混合形状架构。主要区别在于 IMAvatar 侧重于详细的几何形状和外观，而原文的模型更侧重于真实感渲染和高效的训练推理。另一个区别是 IMAvatar 使用向后非刚性光线行进来查找每条光线的规范表面点。

2.2 人脸合成

已经提出了许多用于面部重演和新颖视图合成的方法。基于图像的方法采用扭曲场或编码器-解码器架构来合成图像。由于这些方法表示 2D 空间中的 3D 变形，因此较大的姿势和表情变化可能会出现伪影。可变形模型基于的方法使用参数化 3D 模型来合成数字人像。深度视频人像使用渲染的对应图和图像到图像转换网络来输出照片般真实的图像。延迟神经渲染提出了一种可以由神经渲染器解释的特定于对象的神经纹理。

2.3 神经辐射场 Nerf

NeRF 提出用 MLP 来表示场景，并利用体积渲染来完成新颖的视图合成任务。由于 NeRF 是可微分的，因此它的输入只能是多视图图像。由于上述特性，NeRF 已被广泛应用于 3D 几何重建，4D 场景合成和数字人体建模等此外，很多研究都集中在提高 NeRF 的表示能力上并减少输入数量。近期，NeRF 在人体头部建模方面也展示了其强大的表征能力。许多作品采用 NeRF 来表示动态人体头部场景，并合成高保真 3D 一致的结果。生成头部模型使用潜在代码生成渲染结果。虽然它们通常对结果有很好的姿势控制，但由于其生成对抗性训练策略，不支持表达编辑。通用参数化头部模型将人类头部的潜在空间解构为身份、表达和外观空间，并

在一定程度上实现了对头部变换的语义控制。然而，由于 MLP 容量有限，通用头部模型常常忽略个性化的面部细节和用户特定的面部肌肉运动。AD-NeRF [3] 和 NerFACE 是特定于主题的模型，它可以生成由声音或表情控制的高保真人头动画。然而，AD-NeRF 和 NerFACE 都需要几天的训练和几秒钟的推理。发现他们都倾向于学习平滑的头部场景，有时会忽略高频面部属性。

2.4 NeRF 加速的体素表示

借助体素场，NeRF 可以减轻局部特征的训练负担，这将显着提高训练速度。体素场还可以帮助提前存储密度分布等空间信息，以加快推理速度。最近，即时神经图形基元采用多级哈希表来增强浅层 MLP，并实现了几个数量级的组合加速。它可以用 NeRF 训练一个静态场景，只需几秒钟，并在几十毫秒内渲染场景。然而，由于动态场景复杂的非刚性变形，这些方法不能直接用于动态场景。同时，在动态场景中很难对体素网格进行“修剪”操作，而这通常对于光线行进和信息存储很重要。

3 本文方法

3.1 本文方法概述

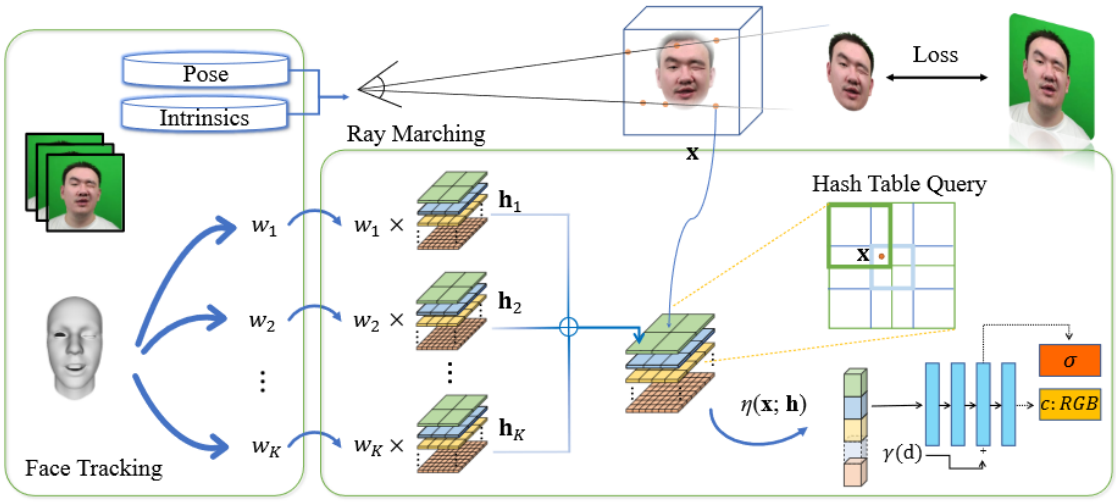

图 1. (1) 跟踪 RGB 序列并获取表达系数、姿势和内在函数。(2) 利用跟踪到的表达式系数来组合多个多级哈希表，得到与特定表达式对应的哈希表。(3) 在哈希表中查询采样点以获得体素特征，使用 MLP 将体素特征解释为 RGB 和密度。修正表情系数并优化哈希表和 MLP 以获得头部模型。

3.2 方法

3.2.1 数据处理

使用现有的基于网格的面部混合形状 Faceware-house 跟踪输入视频中的面部，得到每一帧的表情系数和头部姿态参数。将人体头部姿势参数作为相应帧的外在相机参数，这隐式地

将每个帧的底层几何图形对齐到相同的空间位置。最后，从输入视频中随机提取 N 帧来训练模型，并通过现成的分割方法 Bisenet 生成所选帧的头部掩码。

3.2.2 基于表情的网格更新

使用 128*128*128 密度网格来存储局部密度信息，以指示光线行进跳过空白空间。由于捕获的动态人体头部随着时间的推移而变化，具有不同的姿势和表情，所以不应该用某个表达式的密度场来确定密度网格。在实现中，将每个基础的密度网格计算为：

$$\hat{\mathbf{h}}_i = \mathbf{h}_0 + \hat{w}_i \mathbf{h}_i,$$
$$\text{where} \quad \hat{w}_i = \max_{j \in [1,N]} w_i^j.$$

图 2. 网格更新

3.2.3 基于 NeRF 的线性混合表示

基于 MLP 的隐函数来表示, 将表达式基与多级哈希表相关联并通过精心设计的训练策略赋予每个哈希表指定的语义属性。将模型的表示表示为 R:

$$I = \mathcal{R}_\theta(C, \mathbf{h_0} + \mathbf{Hw}),$$

图 3. 模型表示

3.2.4 渲染过程

首先计算给定表达式系数 w 对应的哈希表为：

$$\mathbf{h} = \mathbf{h_0} + \mathbf{Hw} = \mathbf{h_0} + \sum_{i=1}^{K} w_i \mathbf{h}_i, \mathbf{h} \in \mathbb{R}^{L \times T \times F},$$

图 4. 哈希表

Parameter	Value
Number of levels	16
Hash table size	2^{14}
Number of feature dimensions per entry	4
Coarsest resolution	16
Finest resolution	1024
Initial distribution	$U(-10^{-4}, 10^{-4})$

图 5. 哈希表参数

使用轻量级 MLP 来表示 NeRF 的基于 MLP 的隐函数 $g_(\theta)$ ，$g_(\theta)$ 的网络架构是 4 层 MLP，具有 64 个神经元宽度。表示为：

$$g_\theta : (\eta(\mathbf{x};\mathbf{h}), \gamma(\mathbf{d})) \mapsto (\sigma, c),$$

图 6. MLP 结构

通过体渲染生成渲染图像 I:

$$I(r) = \int_0^\infty p(t)c(r(t))dt,$$
$$\text{where} \quad p(t) = exp(-\int_0^t \sigma(r(s))ds)\sigma(r(t)).$$

图 7. 渲染图像

3.3 损失函数定义

3.3.1 Photometric Loss

颜色损失要求渲染结果与输入 RGB 图像一致，这对于基于 NeRF 的重建很常见。

$$L_{color} = \sum_{r \in S} \|I(r) - I_{GT}(r)\|_1,$$

图 8. 颜色损失。其中 S 是每个批次中的光线集合，I(r)、$I_{GT}(r)$ + 分别是光线 r 的预测 RGB 颜色和真实 RGB 图像的颜色。

3.3.2 Mask Loss

掩码损失。

$$L_{mask} = \sum_{r \in S} \|M(r) - M_{GT}(r)\|_1,$$

图 9. 掩码损失。其中 M(r) 和 $M_{GT}(r)$ 分别是射线 r 的预测掩模值和真实值。这种损失确保头部区域之外的密度为零，并且也让头部区域快速变成不透明物体。

3.3.3 Perceptual Loss

感知损失。选择 VGG 作为 LPIPS 的骨干。由于 LPIPS 使用卷积来提取特征，因此对大小为 W×W 的 B 个补丁进行采样，并在每个批次中渲染总共 B×W×W 光线（B 也是每个批次中的帧数），将渲染的补丁与输入图像上相同位置的补丁进行比较。

3.4 训练过程

$$L_{total} = \lambda_1 L_{color} + \lambda_2 L_{mask} + \lambda_3 L_{LPIPS},$$

图 10. 总体损失

其中 λi 是用于平衡不同项的标量。为了最小化总损失，原文提出了一个精心设计的训练策略，它包含三个步骤：在前两个 epoch。将 λ1、λ2 设置为 1，λ3 设置为 0。掩码损失可以使模型快速学习 3D 密度分布，因为它是直接对密度分布的监督。稳定的密度分布也有利于接下来的密度网格更新。

虽然掩码损失可以帮助密度场的收敛，但发现解析掩码不是很准确，特别是对于头发部分。因此，对于第 2-7 个 epoch，将 λ2、λ3 设置为零并仅使用光度损失。使用 RGB 数据来监督 NeRF 训练，以学习精细的颜色和几何形状。

最后，不仅随机采样光线，还采样 patch。对于随机采样的射线，将 λ1 设置为 1，将 λ2、λ3 设置为 0，对于采样的 patch，将 λ1 设置为 0.1，将 λ3 设置为 0.1。以 1/2 的概率对嘴部 patch 进行采样，否则对整个图像的 patch 进行采样。

4 复现细节

4.1 与已有开源代码对比

通过将网络深度增加到 8 层，发现原先基于 Instant-ngp 的 Fully Fused MLP 结构在处理较大型网络时性能受到限制。为了解决这一问题，我引入了基于 CUTLASS 的 MLP 结构，这在处理更复杂的网络架构时表现更为出色。尽管相较于 Fully Fused 结构而言，新的结构在速度上可能稍慢，但它在保持良好性能的同时应对了更深层次的网络。这一调整确保了模型在更复杂的任务和数据集上具有更强大的表达能力，并且在计算效率上仍然能够保持令人满意的速度。这个结构的选择经过了仔细权衡，确保了性能和速度的双重优势。

```
self.sigma_net = tcnn.Network(
    n_input_dims=sigma_net_begin,
    n_output_dims=1 + self.geo_feat_dim,
    network_config={
        "otype": "CutlassMLP",
        "activation": "ReLU",
        "output_activation": "None",
        "n_neurons": hidden_dim,
        "n_hidden_layers": num_layers,
    },
)
```

图 11. 密度 network

```
self.color_net = tcnn.Network(
    n_input_dims=self.in_dim_color,
    n_output_dims=3,
    network_config={
        "otype": "CutlassMLP",
        "activation": "ReLU",
        "output_activation": "None",
        "n_neurons": hidden_dim_color,
        "n_hidden_layers": num_layers_color ,
    },
)
```

图 12. RGB network

4.2 实验环境搭建

硬件设备：RTX 3090
安装 cuda11.1,Python3.7
安装 pytorch
安装 tiny-cuda-nn
安装 requirements.txt: pip install -r requirements.txt

4.3 使用说明

4.3.1 Train

下载数据集：https://drive.google.com/drive/folders/1OiUvo7vHekVpy67Nuxnh3EuJQo7hlSq1
训练 NeRFBlendShape 模型：bash run_train.sh id1 -500 0
这意味着您要使用 id1 的数据集运行训练，最后 500 帧用于推理，GPU_ID 为 0
调整 run_nerfblendshape.py 的以下选项：
–workspace：实验的工作区文件夹。checkpoints、推理结果和脚本备份都会放在这里。
–basis_num 表情系数的维度为 46。
–to_mem 如果您希望 nerf/provider 在训练之前将整个数据集预加载到内存中，并在每一步中将所需的一批数据从.cpu() 更改为.cuda()，则选择此选项，否则，在每一步中 nerf/provider 将加载首先需要将一批数据从磁盘到内存，然后将数据从 cpu() 更改为 cuda()。
–use_lpips 是否要使用感知损失来改善细节。
–add_mean 将 “1.0” 添加到表达系数中以表示中性表达。

4.3.2 Inference

训练结束后，checkpoint 会被保存在 the_name_of_workspace/checkpoints
根据 checkpoint 进行推理：bash run_infer.sh id1 -500 0 the_path_of_your_checkpoint
使用 generate_video.py 生成视频序列

4.4 创新点

原始论文采用了 4 层 MLP，每层具有 64 个神经元宽度的结构，用于实现三维场景的渲染和重建任务。我将网络的深度增加到 8 层，以提高模型的表示能力和学习复杂特征的能力。通过引入更深的网络结构，期望模型能够更好地捕捉输入数据中的抽象特征，并提高对场景密度和颜色的准确建模能力。这种创新不仅有助于改善模型的性能，而且可能带来更具表达力的特征学习，从而增强了模型在复杂三维环境中的推断能力。

5 实验结果分析

本部分对实验所得结果进行分析，详细对实验内容进行说明，实验结果进行描述并分析。通过增加网络深度，得到了相较于原文 PSNR 值 (34.15) 略高一些的值

```
PSNR:34.3078:
```

图 13. PSNR:the higher the better

图 14. ours

6 总结与展望

本部分对整个文档的内容进行归纳并分析目前实现过程中的不足以及未来可进一步进行研究的方向。在本研究中，我们通过改进神经辐射场（NeRF）方法，实现了个性化的人头表示，取得了略优于原文的结果。通过引入多级体素场与潜在空间中的表情系数相结合，我们成功地学习了不同尺度的头部细节，使得生成的人头图像更具真实感。线性混合设计和局部特征调节进一步提高了面部细节的逼真程度。在训练模型时，我们采用了精心设计的训练策略，包括光度损失、掩模损失和知觉损失的组合，以最大限度地减少总损失。

未来的研究方向可以包括进一步优化模型结构和训练策略，以提高生成图像的质量和逼真程度。可以探索更复杂的神经网络架构，或者引入先进网络结构，以增强模型的表达能力。此外，可以考虑在更广泛的数据集上进行实验，以验证模型在不同场景和不同个体上的泛化能力。另一方面，我们可以关注模型的实时性能，进一步提高交互式渲染速度。这可能涉及到对模型的推理过程进行优化，或者探索硬件加速的可能性。在实际应用方面，可以考虑将该方法扩展到其他领域，如虚拟现实、增强现实或人机交互，为用户提供更丰富和个性化的体验。

参考文献

[1] Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425, 2013.

[2] Guy Gafni, Justus Thies, Michael Zollhofer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8649–8658, 2021.

[3] Yudong Guo, Keyu Chen, Sen Liang, Yong-Jin Liu, Hujun Bao, and Juyong Zhang. Ad-nerf: Audio driven neural radiance fields for talking head synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5784–5794, 2021.

[4] Yang Hong, Bo Peng, Haiyao Xiao, Ligang Liu, and Juyong Zhang. Headnerf: A real-time nerf-based parametric head model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20374–20384, 2022.

[5] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[6] MA TURK. A morphable model for the synthesis of 3d faces. In *SIGGRAPH'99*, 1999.

[7] Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C Bühler, Xu Chen, Michael J Black, and Otmar Hilliges. Im avatar: Implicit morphable head avatars from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13545–13555, 2022.