

# NeRFPlayer: 分解神经辐射场表示流媒体动态场景

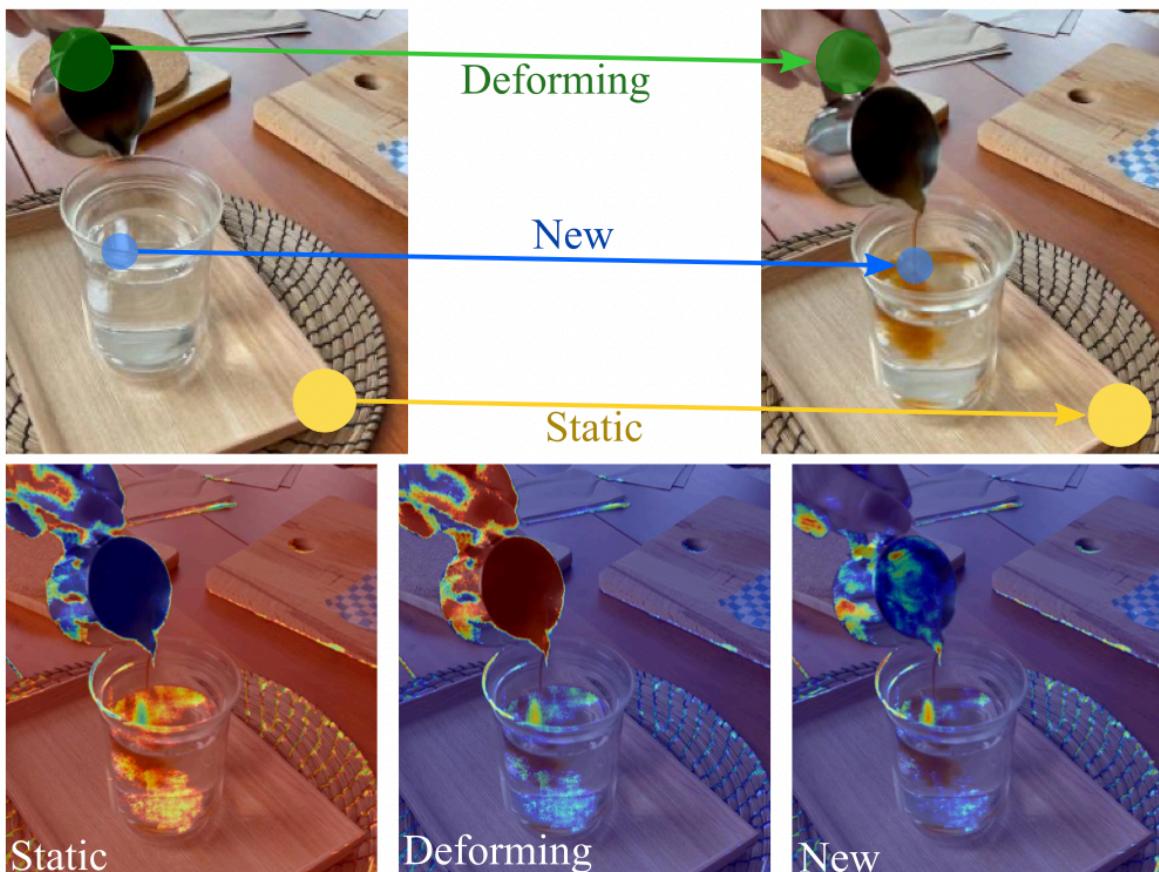
本文提出了动态场景的三维表征方法：NeRFPlayer，该方法通过将神经辐射场分解为表示多种时域状态的功能模块，将时间维度的信息引入三维场景的重建中，实现四维时域空域神经表征，该方法还创新性地提出了针对特征的滑动窗方案以适应流媒体播放的需求。作者通过实验验证了所提出了分解场的合理性，并且在单相机移动拍摄动态场景数据集和多相机拍摄动态场景数据集上进行实验，与现有动态场景 NeRF 学习方法进行对比，获得 SOTA 表现。

## 简介

NeRF 在静态场景中取得了成功，但将其扩展到处理动态场景仍然具有挑战性。由于以下两个原因，将额外的时间维度  $t$  引入 NeRF 的 5D 表示并非易事：

1. 时空点  $(x, y, z, t)$  的监控信号比静态点  $(x, y, z)$  稀疏。静态场景的多视图图像很容易获取，因为我们可以在四处移动相机，但动态场景中的额外视图需要额外的记录相机，导致输入视图稀疏。
2. 场景的外观和几何频率沿空间轴和时间轴是不同的。当前景物体移动置时，内容通常会发生很大变化，但背景场景不太可能发生较大变化。时间  $t$  维度的不适当频率建模会导致较差的时间插值效果。

针对上述两个问题，现有的解决方案包括采用运动模型来匹配点或利用深度和光流等数据驱动的先验。与现有工作不同，我们认为动态场景中不同的空间区域具有不同的时间特征。我们假设动态场景中存在三种时间模式：静态（static）、变形（deforming）和新区域（new）。因此，我们将动态场景分解为上述三类类别，每个类别有对应的分解场，这是通过空间点属于每个类别的概率来实现的。分解场使用自监督方法训练，通过手动设置的全局简单正则化进行约束。



overview

我们的方法如何解决建模四维时域空域 NeRF 的难点：首先，为每个分解区域引入不同的时间正则化，从而减轻从稀疏观察重建的模糊性。例如，静态区域分解将动态建模简化为静态场景建模问题。变形区域强制前景对象在动态场景中保持一致。其次，根据场景的时间特性将场景分成不同的区域，使得每个区域在时间维度上的频率一致。

针对空间频率和时间频率之间的差异，我们使用了混合表征方式 (instant-ngp, tensorRF 等)。我们没有设计  $(x, y, z, t)$  特征体积的网格，而是将  $(x, y, z)$  特征体积的通道视为时间相关的。为了支持可流式动态场景表示，我们在特征通道上提出了一种滑动窗口方案，以将  $t$  引入表示中。

本工作的贡献可以总结为以下四点：

- 我们提出根据动态场景的时间特征来分解动态场景。通过分解场来实现，分解场将每个  $(x, y, z, t)$  点作为输入，并输出属于三类的概率：静态 (static)、变形 (deforming) 和新区域 (new)。
- 我们设计了一种自监督方案，用于优化分解场并使用全局的简约的损失函数对分解场进行正则化。
- 我们在NeRF的混合表征上设计了一个滑动窗口方案，用于有效地建模时空场。
- 单相机移动拍摄动态场景数据集和多相机拍摄动态场景数据集上进行了大量实验和实时渲染演示。我们的消融实验证了所提出的三种时间模式的必要性。

## 方法

---

### 先导知识

对于 NeRF 中的每个点  $p=(x,y,z)$ ，我们用体积渲染在射线上表示它。设相机的光学原点和方向为  $o$  和  $d$ ，然后通过  $p_i = o + id$  对一组点进行采样，并通过下式计算预期颜色  $C(r)$ ：

$$C(\mathbf{r}) = \int_{i_n}^{i_f} e^{-\int_{i_n}^i \sigma(\mathbf{p}_j) dj} \sigma(\mathbf{p}_i) \mathbf{c}(\mathbf{p}_i, d) di$$

在原始 NeRF 中，辐射场由 MLP 隐式表示，该 MLP 将点  $p$  作为输入并输出其密度和颜色。然后使用重建颜色和地面真实颜色  $C_{gt}(\mathbf{r})$  之间的重建损失训练 MLP，即：

$$L_{rec} = \sum_{\mathbf{r} \in \mathcal{R}} \|C(\mathbf{r}) - C_{gt}(\mathbf{r})\|_2^2$$

NeRF 中的隐式表示非常紧凑但计算量大，导致训练和渲染速度慢。近期工作提出了显式和隐式的混合表示，以有效地重建和渲染辐射场。尽管这些方法有其独特的优势，但所有这些混合表示都遵循一个共同的框架。部分工作显式地存储特征  $V$ ，可以是体素网格、哈希表或一组基向量矩阵的形式。对于 3D 空间中的任何点  $P$ ，可以通过低计算成本的操作（例如，体素的三线性插值）有效地获得特征向量  $up = V(p)$ 。接下来，采用解码器  $D$  从  $up$  中获取点的密度和颜色  $c$  等属性。解码器  $D$  可以是 MLP 或球谐函数。

### 分解时域空域表征

我们的方法建立在动态场景中的不同区域可以具有不同的时间变化模式的假设之上。使用不同的时间正则化对不同区域进行建模不仅有助于保持时间一致性，还可以节省计算量。例如，背景中的某些对象可能在动态序列中具有静态几何形状，这使我们能够降低其表示的容量和复杂性。我们方法对不同的动态区域进行分类，然后根据它们的类别使用不同的表示对不同的动态区域进行建模。

我们假设动态场景中存在三种区域，并使用单独的流程对这些区域进行建模：

- **静态 (static)** 区域在动态场景（例如桌子）中具有恒定的几何形状和位置。此外，我们假设静态区域的外观不会随时间频繁变化，即在时间上是低频的。这是因为场景中静态部分的外观变化主要由照明条件的改变引起，反照率是时不变的。因此，静态域  $s(\cdot)$  用于表示静态点。
- **变形 (deforming)** 区域为具有变形表面的对象建模，例如图2中的手和杯子，变形区域可能包括刚性或非刚性运动。变形点由变形场  $d(\cdot) : (\mathbf{p}, t) \mapsto (\Delta\mathbf{p})$  表示。然后将变形点  $\mathbf{p} + \Delta\mathbf{p}$  作为查询点发送到预定义的空间场（例如，静态域  $s$ ）。
- **新 (new)** 区域模拟序列中某个点出现的新内容，例如将浓缩咖啡倒入水中后的新液体。采用输入为  $(\mathbf{p}, t)$  的新网络  $n(\cdot)$  来学习和表示新区域。

为了分解场景，我们设计了一个分解场  $f(\cdot) : (\mathbf{p}, t) \mapsto (P_{\text{static}}, P_{\text{deform}}, P_{\text{new}})$ ，其中  $P_{\text{static}}$ ,  $P_{\text{deform}}$ ,  $P_{\text{new}}$  表示静态、变形和新的概率。接下来，我们将上述网络  $(s, n)$  的输出视为特征向量而不是点密度等属性，并将输出特征向量分别表示为  $v_{\text{static}}$ 、 $v_{\text{deform}}$ 、 $v_{\text{new}}$ 。最后，给定一个查询点  $\mathbf{p}$ ，我们首先收集上述网络的输出，然后计算该点的最终特征向量  $\mathbf{v} = \sum_* P_* v_*$ 。然后  $\mathbf{v}$  被输入到一个轻量级的网络进行密度和颜色预测。

在下图中，我们使用一个简单的 2D 示例演示了我们的方法。图中的任务是从给定的 2D 图像进行时间插值。上面提到的字段将  $(x, y)$  位置作为输入。字符串“2022”进行刚性移动，而字符串“VR”逐渐出现。不同的插值性能证明了对具有变形场和新场的动态场景进行建模的必要性。

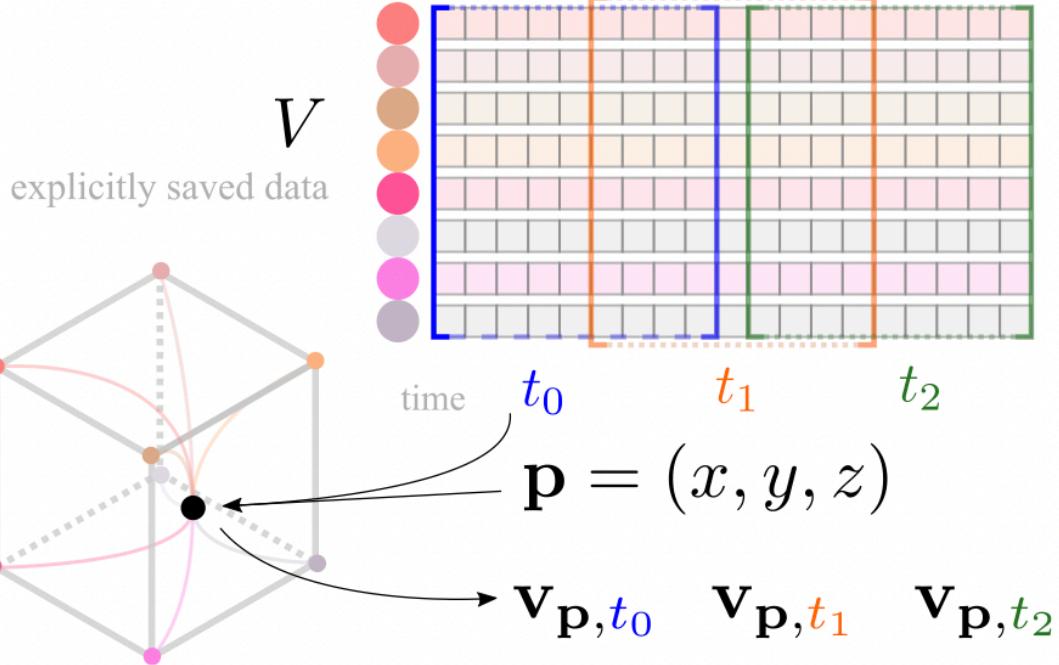
	2022	2022	2022	2022	2022	2022	2022	2022	2022
inputs	2022	2022	2022	2022	2022	2022	2022	2022	2022
w/o defrom	2022	2022	2022	2022	2022	2022	2022	2022	2022
w/o new	2022	2022	2022	2022	2022	2022	2022	2022	2022
full	2022	2022	2022	2022	2022	2022	2022	2022	2022

2D 示例

## 流式混合表征

我们观察到显式表示  $V$  通常由具有预定义特征维度的数组  $\text{entry}$  组成。例如，Instant-NGP 中哈希表中的每个  $\text{entry}$  和 TensoRF 中的每个基向量/矩阵都具有固定的特征维度。因此，我们提出流式传输特征通道，以便  $V$  可以是从时空点  $(\mathbf{p}, t)$  到固定长度特征向量的映射。

我们使用具有滑动窗口的特征通道以及时间维度  $t$ ，如下图所示。假设对于每一帧，特征向量  $\mathbf{v}_{p,t}$  的维度为  $F$ ，并且  $k$  个通道是新的新需求帧，则对于  $T$  帧序列，数组条目  $v \in V$  的维度为  $F + k(T - 1)$ 。对于单帧  $t$ ， $V$  中的通道  $[kt, kt + F]$  将用于计算  $\mathbf{v}_{p,t}$ ，例如 InstantNGP 中的三线性插值或 TensoRF 中的张量乘法。



具有滑动窗口的特征通道

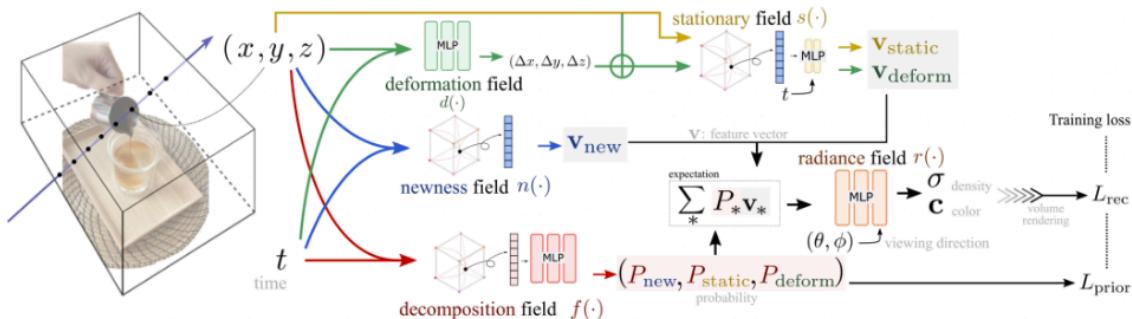
为了确保  $up,t$  与  $t$  一起平滑转换，对特征通道进行了重新排列以匹配共享通道。例如，设  $t = 0$ 、 $k = 2$  和  $F = 4$ ，则  $V$  的通道  $[0, 1, 2, 3]$  用于  $up,0$ 。接下来，我们对  $t = 1$  使用通道  $[4, 5, 2, 3]$ ，对  $t = 2$  使用通道  $[4, 5, 6, 7]$ 。重新排列背后的原则是共享特征通道始终对齐向量中的相同索引。否则，无法保证帧之间的平滑转换。

流媒体通道很容易使我们能够通过线性插值特征向量在两个观察到的帧  $ts$  和  $ts+1$  之间临时插值帧：

$$\mathbf{v}_{p,t} = \frac{t - ts}{ts+1 - ts} \mathbf{v}_{p,ts+1} + \frac{ts+1 - t}{ts+1 - ts} \mathbf{v}_{p,ts}$$

# 整体框架

下图是算法的整体流程框架。分解场  $f$  由显式缓存的特征（表示为  $V_f$ ）和一个小型 MLP 解码器  $D_f$  组成。变形场  $d$  是一个 MLP，因为变形是稀疏的并且是低频的，使用小型 MLP 即可。静态场由显式缓存的特征（用  $V_s$  表示）和一个微型 MLP 解码器组成。时间  $t$  和从  $V_s$  获得的特征将作为微型 MLP 的输入。使用微型 MLP 的原因是为了对由时变照明引起的时间相关外观变化进行建模，我们假设照明是低频的。新区域场是显式保存的特征  $V_n$ 。请注意，在上面的显式表示中， $V_f$  和  $V_n$  都采用 4D 输入  $(p, t)$ ，因此这里使用流媒体通道。然后通过辐射场  $r$  对最终的预期特征向量  $v$  进行解码。与 NeRF 一样，观察方向  $(\theta, \varphi)$  也被发送到  $r$ 。



算法的整体流程框架

## 数据集介绍

本方法在一个单相机移动拍摄动态场景数据集和两个多相机拍摄动态场景数据集上进行验证。

- **Immersive Video** 包含了 46 路同步拍摄的 4k 鱼眼相机，作者提供的原始视频数据包含了不同的相机设置参数，例如曝光度和白平衡，我们挑选了 7 个拥有相似图片参数的动态场景，在实验中，我们将图片分辨率调整为  $1280 \times 960$ 。
- **Plenoptic Video** 是 21 台相机在  $2704 \times 2028$  分辨率下拍摄的。与主要在室外场景拍摄的 Immersive Video 不同，Plenoptic Video 包含不同光照条件的室内场景，在实验中我们将分辨率降到  $1352 \times 1014$ 。
- **HyperNeRF** 仅提供了每个时间戳每个场景下的一个相机视角，我们使用的设定： $960 \times 540$  用于质量评估， $1920 \times 1080$  用于量化评估。

## 与现有工作的对比

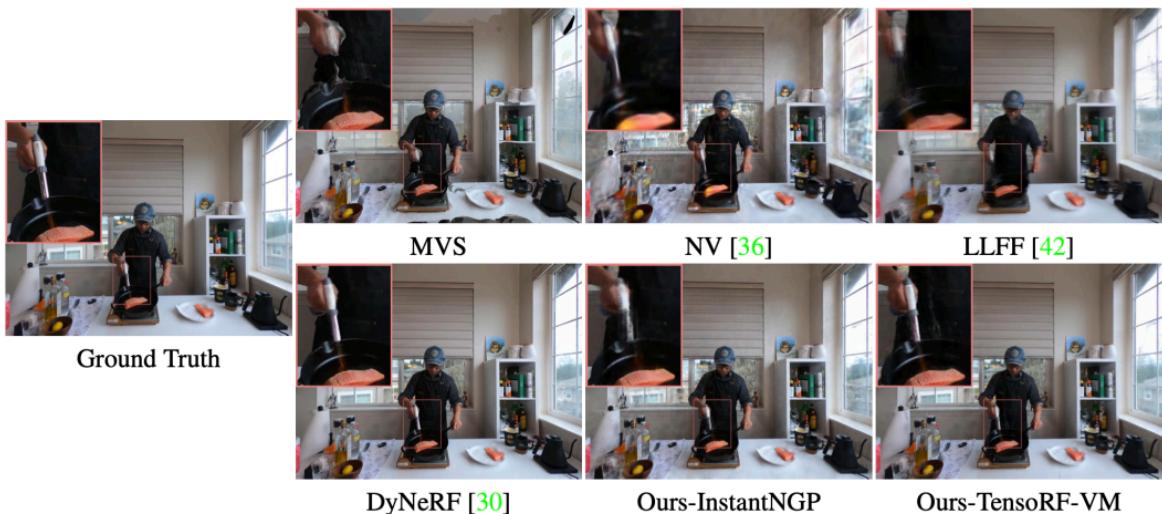
### 多相机数据

我们分别以 Instant-ngp 和 tensorRF 为 backbone 与现有方法进行对比，表 1 给出了量化对比的结果，我们在训练和渲染效率远高于对比工作的基础上，获得了更好的质量效果。

Method	PSNR↑	Training Time (GPU Hours)	Rendering Time (s/img)
Neural Volumes [36]	22.797	-	-
LLFF [42]	23.238	-	-
NeRF-T [30]	28.448	-	90
DyNeRF [30]	29.580	1344	90
Ours-InstantNGP	30.293	5.5	10.8
Ours-TensoRF-VM	30.692	6	22.1

表1 量化对比的结果

下图展示了我们方法的渲染效果，我们方法在建模动态场景中快速移动的目标结果更加精细。



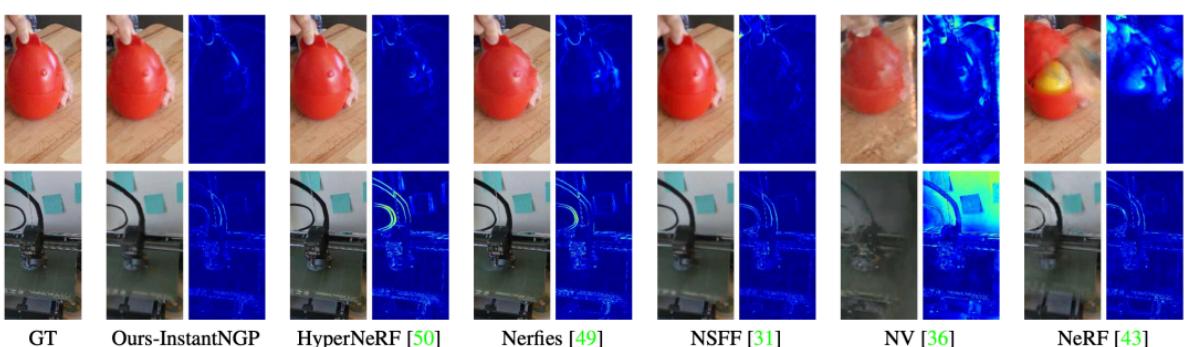
渲染效果对比

**单相机数据** 表 2 中我们的方法与 SOTA 单相机重建方法进行对比，其中 NSFF 依赖数据驱动的深度和光流作为先验。

Method	Rendering Time (s/img)	Broom		3D Printer		Chicken		Peel Banana		Mean	
		PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
NeRF [43]	~75	19.9	0.653	20.7	0.780	19.9	0.777	20.0	0.769	20.1	0.745
NV [36]	<0.03	17.7	0.623	16.2	0.665	17.6	0.615	15.9	0.380	16.9	0.571
NSFF [31]*	~90	26.1	0.871	27.7	0.947	26.9	0.944	24.6	0.902	26.3	0.916
Nerfies [49]	~90	19.2	0.567	20.6	0.830	26.7	0.943	22.4	0.872	22.2	0.803
HyperNeRF [50]	~90	19.3	0.591	20.0	0.821	26.9	0.948	23.3	0.896	22.4	0.814
Ours-InstantNGP	4.8	21.7	0.635	22.9	0.810	26.3	0.905	24.0	0.863	23.7	0.803

表2 与单相机重建对比

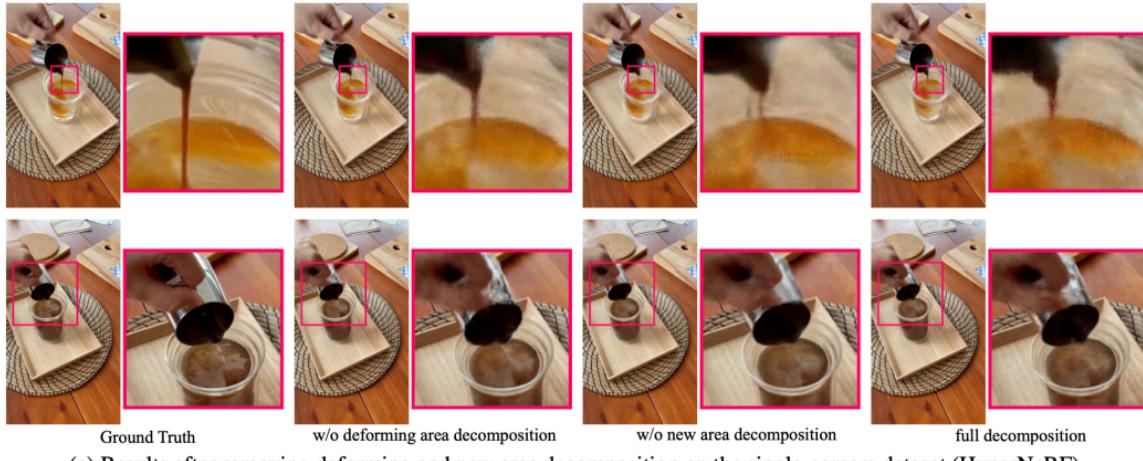
下图为上述工作视觉效果的质量对比。



视觉效果的质量对比

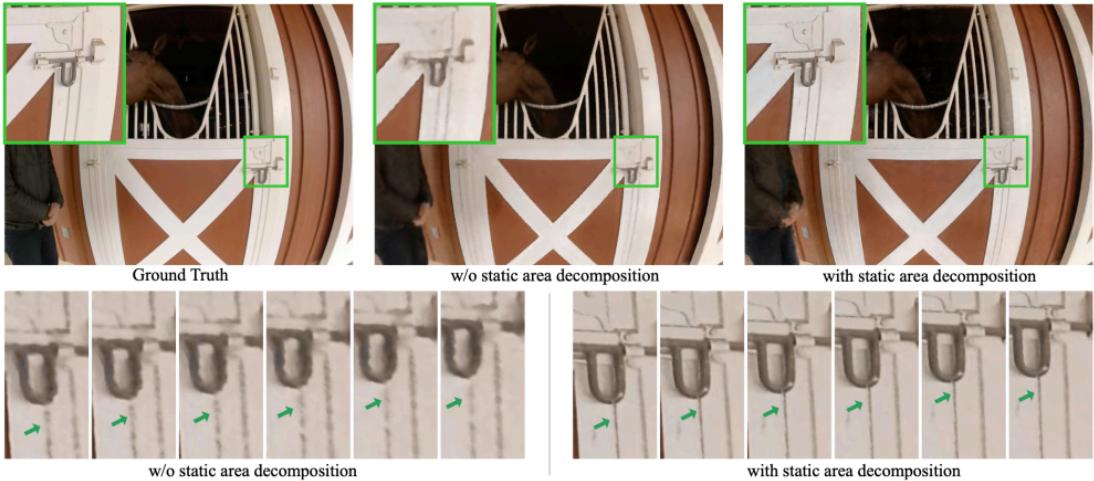
## 消融实验

我们可以从下图中观察到，去除新区域分解会导致新倒出的浓缩咖啡建模失败，去除变形区域分解会导致手和杯子模糊。



(a) Results after removing deforming and new area decomposition on the single-camera dataset (HyperNeRF).

其次，我们研究了我们的分解对多相机数据集的影响。下图表明静态区域在分解静态区域后变得更清晰。此外，在没有静态区域分解的情况下，我们观察到当我们以新的时间和视角渲染图像时背景会闪烁。



(b) Ablation of static area decomposition on the multi-camera dataset (Immersive Video). Second row: novel time and view rendering.

最后，我们研究了变形区域分解对多相机数据集的影响。本实验的目的是观察到使用和不使用变形场建模的渲染几乎没有差异（PSNR 差异小于 0.1）。我们假设这是因为帧之间物体的运动对于具有高 FPS 记录速率的相机来说很小。因此，用 新区域场对所有动态区域进行建模仍然可以产生平滑的插值。手动下采样帧采样率进行训练，以放大帧之间的运动。我们可以观察到，在没有变形建模的情况下，在两个训练时刻之间进行插值时，移动头盔首先消失然后重新出现。作为比较，如果对变形进行建模，头盔的内容会得到很好的保留。



(c) Ablation of deforming area decomposition on the multi-camera dataset (Immersive Video). Every 8 frames are used for training.

