更快、更小、更美观：用于二维高斯视频编码的 Fast-GSVC

Longan Wang 新加坡国立大学 新加坡
wanglongan@u.nus.edu

Yuang Shi 新加坡国立大学 新加坡
yuangshi@u.nus.edu

Chia Qian Tan 新加坡国立大学 新加坡
tanchiaqian@gmail.com

Wei Tsang Ooi*新加坡国立大学 新加坡
dcsooiwt@nus.edu.sg

arXiv:2611.30629v1 [cs.MM] 23 Nov 2026

摘要

二维高斯已成为一种颇具前景的视频显式表示方法。然而，将高斯拟合到视频序列需要大量计算，导致编码时间较长。在本文中，我们提出了 Fast-GSVC，这是一种二维高斯视频编解码器，可在提升率失真性能的同时加速编码。我们确定了二维高斯泼溅训练中的关键瓶颈，并通过改进初始化、更平滑但更激进的参数更新、由粗到细的渐进式高斯优化，以及用于高斯初始化和时间传播的光流引导来加速优化。我们还发现，现有编解码器的量化微调过程未能充分利用帧间的时间一致性，导致压缩效果欠佳。为此，我们保留时空相关性以维持集中的残差，并引入时序感知量化，进一步提升率失真性能。在 UVG 数据集上的实验中，Fast-GSVC 最多可将编码时间缩短 92.5%，相当于实现了相对于 GSVC 的 13.4× 倍加速比。在峰值信噪比相当的情况下，文件大小最多可减少 58%；在每像素比特数相当的情况下，峰值信噪比最多提升 2.5 分贝。项目页面：https://longanwang-cs.github.io/fastgsvc/。

CCS 概念

• **计算方法** → **图像压缩**。

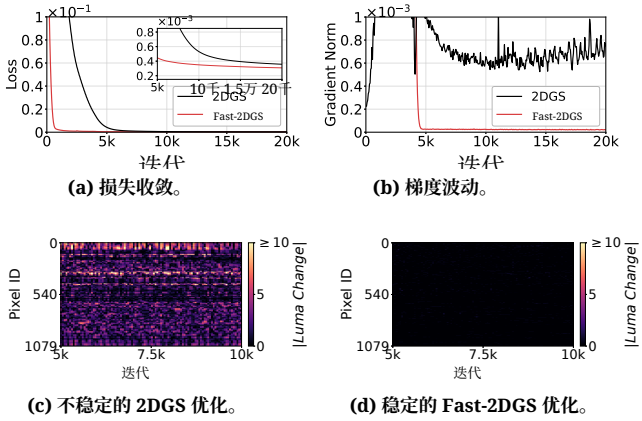
关键词

二维高斯泼溅、视频编码、系统加速

1 引言

几十年来，视频编码一直依赖于固定像素网格上的信号处理，例如 AVC [34], HEVC [26], 和 VVC [4]等标准。除了这一范式之外，人们也探索了神经方法，将视频编码为网络参数或潜在表示 [10–12, 23]。尽管这类表示较为紧凑，但其隐式表示在解码时需要进行神经推理，限制了精细控制和直接编辑。近年来，三维高斯泼溅（3DGS） [8] 作为一种颇具前景的视频显式表示 [24, 27]应运而生。由于高斯可直接访问和操控，3DGS 支持

快速渲染 [8],任意分辨率投影 [37],直接编辑 [24, 27],用于分层流媒体的渐进式重建 [21],



(a) 损失收敛。 (b) 梯度波动。

(c) 不稳定的 2DGS 优化。 (d) 稳定的 Fast-2DGS 优化。

图 1：2DGS 与 Fast-2DGS 优化动态的比较。(a) Fast-2DGS 能更快地降低损失，插图进一步显示 Fast-2DGS 收敛到更低的损失。(b) Fast-2DGS 减少了梯度波动。(c,d) 沿中央垂直扫描线，在采样的优化迭代次数之间逐像素亮度变化的绝对值，显示 Fast-2DGS 中的像素波动显著减少。

以及灵活的尺寸—质量控制 [22]。然而，由于致密化后参数量庞大，以及深度排序阿尔法混合的高昂成本，3DGS 的计算开销很大。

2024 年，二维高斯泼溅（2DGS） [39] 被引入用于图像表示与压缩。通过直接在图像平面上运行，2DGS 消除了 3DGS 所需的深度和视角相关建模，将每个高斯的参数数量从 59 个减少到 8 个，并以低成本的累积求和取代了深度排序阿尔法混合。在这些优势的基础上，GSVC [30] 引入了首个基于 2DGS的高斯视频编解码器。为了利用时间一致性，GSVC 使用锚帧和传播帧表示视频。锚帧首先从头开始独立优化，类似于帧内编码帧；而传播帧则通过继承并细化前一帧中的高斯来利用帧间依赖。高斯可以被添加和移除，以适应动态变化和冗余；场景变化则会触发新的锚帧。在压缩方面，

*通讯作者



(a) GSVC（2DGS）。



(b) Fast-GSVC（Fast-2DGS）。

图 2：在 Jockey 上使用 10千个高斯进行训练时，前 4千次迭代的训练动态，峰值信噪比按相应训练分辨率测量。经过 4千次迭代后，Fast-GSVC 已达到 34.2 分贝，并生成拟合良好的重建结果；而 GSVC 仅达到 18.2 分贝，仍存在明显的孔洞和伪影。

GSVC 采用结合残差编码的量化感知微调。这些设计利用时间冗余实现紧凑表示，同时达到业界领先的率失真性能。然而，即使使用轻量级 2DGS 和 H100 GPU，GSVC 编码每个 1080p 帧仍需要数百秒，累积起来，编码一段 600 帧视频需要超过四天。

为了解 2DGS 优化速度缓慢的原因，我们检查了其迭代优化过程中的损失曲线，如图 1a 所示。该过程分为两个阶段：首先是初始拟合阶段，高斯不断扩展以填充未覆盖区域，损失下降相对较快；随后进入细化阶段，优化持续细化细节，但损失下降速度较慢，如插图所示。进一步分析这两个阶段后，我们发现了 2DGS 优化中的两个关键瓶颈：(i) 初始拟合阶段需要大量迭代才能建立覆盖，如图 2a 所示；(ii) 后续细化仍不稳定，图 1b 中的梯度波动较大，图 1c 中可见明显闪烁，导致收敛变慢。这些观察表明，除了底层 GPU 加速之外，还可以通过提高优化效率来进一步加速基于 2DGS 的高斯视频编码，以更少的迭代次数达到更高或相当的质量。

我们首先通过覆盖感知初始化和激进参数更新来解决这些瓶颈，从而加速初始拟合，并减少未覆盖区域，如图 2b 所示。对于后续细化，历史参数平均能够抑制短期波动，同时保持激进更新，如图 1b 和图 1d 所示。这些技术相结合，实现了更快的收敛（如图 1a 所示），并获得了更低的重建损失（如插图所示）。

接下来，我们通过针对锚帧从头训练、传播帧连续优化以及面向量化的压缩微调而定制的设计，将我们的方案扩展到视频编码。对于锚帧，Fast-GSVC 通过由粗到细重建，逐步提高目标复杂度和计算成本，从而简化从头开始的优化。对于传播帧，Fast-GSVC 通过光流以两种方式进一步利用时间信息：额外的高斯被引入无法通过基于光流的传播可靠解释的动态区域，从而提供更好的初始化；在优化过程中，光流作为运动先验，引导从前一帧传播而来的高斯沿估计运动方向移动，加速收敛。对于压缩，Fast-GSVC 保留时空相关性，以维持集中的残差，并引入时序感知量化，进一步提升率失真性能。最后，Fast-GSVC 通过 CUDA图 在整个流水线中优化图形处理器执行，减少优化过程中的重复内核启动开销。我们的贡献总结如下：

- 我们通过改进初始化、更平滑但更激进的参数更新、由粗到细的渐进式优化，以及光流引导的初始化与传播，加速 2DGS 编码。
- 我们通过时序感知量化和时空约束，在保持时间连贯性的同时提升率失真性能。

在 UVG 上进行的大量实验表明，Fast-GSVC 将编码时间缩短了 92.5%，实现了相较于 GSVC 的 13.4× 倍加速比。在质量相当时，每像素比特数降低约 58%；在每像素比特数相当时，峰值信噪比提高约 2.5 分贝。

2 相关工作

2.1 迭代式高斯泼溅加速

高斯泼溅 [8], 最初以 3DGS 的形式提出，通过迭代优化重建静态三维场景。现有的 3DGS 加速方法主要通过降低每次迭代的计算量或改善收敛来实现加速。Taming 3DGS [16]、FastGS [20],和 Group Training [29] 通过控制活跃高斯的数量或选择性地更新基元来减少计算量。DashGaussian [5]进一步采用渐进式分辨率和高斯预算，以避免不必要的计算；而 QuickSplat [15] 则通过改进初始化和致密化来加速收敛。这些方法主要面向三维场景和表面重建，在渲染过程中需要对数百万个高斯进行投影、深度排序和阿尔法

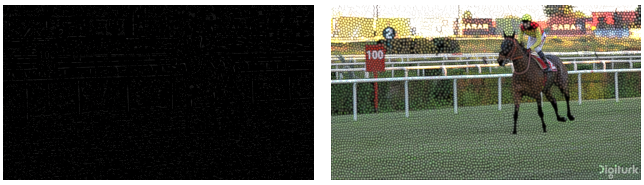
混合，因此每个优化步骤的计算成本都很高。相比之下，轻量级二维高斯泼溅 [39] （2DGS）专为图像 [13, 39] 和视频 [9, 30] 重建而开发，直接在图像平面上运行，使用更加紧凑的二维高斯，从而避免成本高昂的三维投影和深度排序。尽管每次迭代的成本较低，2DGS 优化仍然效率不高，拟合速度慢且细化过程不稳定。在本工作中，除了底层执行加速之外，我们还通过提高优化效率来加速二维高斯泼溅训练，使其能够以更少的步骤达到相当或更优的解。

2.2 视频编码

信号处理长期以来一直是视频编码的基础。它通过帧内/帧间预测和运动补偿等人工设计的模块，利用空间和时间冗余，并进一步使用离散余弦变换（DCT）等变换压缩预测残差。经过数十年的发展，这些技术已演变为 H.264/AVC [34], H.265/HEVC [26], 和 H.266/VVC [4]等广泛采用的标准。与此同时，业界也探索了其他替代方法。受深度学习发展的推动，神经网络方法（例如， [10–12, 23]）使用学习到的模型参数或潜在表示对视频进行隐式表示，并取得了颇具潜力的率失真性能。在本文中，我们探索了另一种基于二维高斯泼溅的替代方案，该方案使用显式且可直接访问的高斯基元表示视频内容。

2.3 高斯视频

高斯泼溅已成为视频重建与处理的一种显式表示，VGR [27] 和基于 3DGS 的 VeGaS [24] 对此进行了验证。近期，轻量级 2DGS 沿两个主要方向扩展到视频领域。一个方向聚焦于可泛化的视频重建。GaussianVideo [9]通过随时间变形二维高斯来建模时间变化，GVT [6] 利用学习到的时空特征生成高斯令牌，而 HyperGS [41] 使用前馈模型预测逐帧高斯表示。另一个方向采用逐视频迭代优化进行视频编码，自然支持灵活的率质量控制：视频可以使用任意高斯预算进行优化，以实现码率控制；同时，渐进式剪枝可以满足目标质量，从而实现恒定质量编码 [30]。沿着这一范式，GSVC [30] 引入了首个二维高斯视频编解码器，集成时间传播、连续学习、量化、熵编码、场景变化检测以及高斯添加/移除，以适应视频动态。借助对高斯预算的细粒度控制，P-GSVC [32]进一步通过渐进式 2DGS 扩展 GSVC，实现可伸缩分层编码，分层开销低于 3%。然而，这种率质量灵活性需要大量迭代优化，从而导致较长的编码时间。在 Fast-GSVC 中，我们在保留细粒度率质量控制的同时，加速端到端高斯视频编码，并进一步提升压缩性能。我们还通过初始化研究了前馈预测这一潜在的加速路径，相关分析见第 4节。



(a) 不使用表示阶段 CI　　(b) 使用表示阶段 CI

图 3：使用和不使用表示阶段 CI 时，经过 250 次迭代后的重建结果。

3 Fast-GSVC

3.1 问题表述

给定一个二维视频 $\mathrm{F} = \{\boldsymbol{F}_t\}_{t=1}^{T}$，高斯视频编解码器将˜每一帧编码为紧凑表示 G_t，并据此获得重建帧，如下所示

$$\hat{\mathcal{F}} = \{\hat{\boldsymbol{F}}_t\}_{t=1}^{T} = \{\mathcal{D}(\tilde{\mathcal{G}}_t)\}_{t=1}^{T}, \tag{1}$$

其中，D(·)表示解码过程。对于 $\boldsymbol{F}_t$，我们的目标是提高优化效率，使其能够通过更少的迭代次数达到更优解。然而，每次迭代的实际计算量会随高斯函数数量及其重叠情况而变化。因此，我们将墙上时钟时间作为衡量优化效率的实际指标： $T^t_{\text{opt}} = \sum_{k=1}^{K_t} \tau_{t,k}$，其中 K_t表示帧 $\boldsymbol{F}_t$的优化迭代次数，$\tau_{t,k}$ 表示其第 k次迭代的运行时间。因此，我们的目标是在保持目标率失真性能的同时，最小化逐帧优化时间：

$$\text{最小化}\ T^t_{\text{opt}} \quad \text{s.t.} \quad \mathrm{L}(\hat{\boldsymbol{F}}_t, \boldsymbol{F}_t) \le \mathrm{L}^{\text{target}}, \qquad B_t \le B_{\text{target}}, \tag{2}$$

其中，L(·,·)表示两帧之间的重建损失，B_t 表示编码帧大小，$\mathrm{L}^{\text{target}}$ 和 B_{target} 根据 GSVC 经过大量迭代优化后收敛的性能，共同定义目标率失真工作点。

3.2 Fast-2DGS

2DGS 表示。 我们采用轻量级 2DGS 作为表示基元。以帧 $\boldsymbol{F}_t$ 为例，为简洁起见，省略帧和高斯索引，并将 $G_i^t = (\boldsymbol{\mu}_i^t, \Sigma_i^t, \boldsymbol{c}_i'^t)$ 写作 $G = (\boldsymbol{\mu}, \Sigma, \boldsymbol{c}')$，其中 $\boldsymbol{\mu} \in \mathbb{R}^2$ 表示图像平面上的高斯中心，$\Sigma \in \mathbb{R}^{2\times2}$ 控制其形状和方向，$\boldsymbol{c}' \in \mathbb{R}^3$ 表示不透明度加权的RGB颜色，其中不透明度被吸收到颜色系数中。遵循 GaussianImage [39],，这些系数不受 $[0, 1]$, 限制，从而允许高斯同时添加和移除内容。为保证协方差矩阵为半正定矩阵，我们使用 Cholesky 分解对 $\Sigma = \boldsymbol{L}\boldsymbol{L}^\top$ 进行参数化，其中 $\boldsymbol{L}$ 是一个由 Cholesky 向量 $\boldsymbol{l} = [\ell_0, \ell_1, \ell_2]^\top$ 参数化的下三角矩阵。因此，每个高斯包含八个参数：两个用于位置，三个用于形状，三个用于颜色。在没有深度排序的情况下，2DGS 将每个像素重建为相互重叠高斯的加权和。对于像素 j，其重建颜色为

$$\boldsymbol{C}_j = \sum_{n\in\mathcal{N}_j} \boldsymbol{c}'_n \exp\left(-\frac{1}{2}\boldsymbol{d}_n^\top \Sigma_n^{-1} \boldsymbol{d}_n\right), \tag{3}$$



（a）不含量化阶段覆盖感知初始化　（b）含量化阶段覆盖感知初始化

图 4：含量化阶段覆盖感知初始化和不含量化阶段覆盖感知初始化的初始渲染结果，前者能够改善空间覆盖。

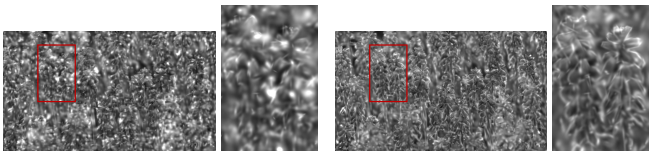
其中，$\mathrm{N}j$ 表示对像素 j 有贡献的高斯集合，$\boldsymbol{d}_n$ 是像素 j 与高斯 n 中心之间的位移。高斯表示通过可微迭代优化获得，

$$\mathcal{G}_t = \arg\min_{\hat{\mathcal{G}}_t} \mathcal{L}\big(\mathcal{R}(\mathcal{G}_t), \boldsymbol{F}_t\big), \tag{4}$$

其中，R(·) 表示二维高斯渲染。正如第 1 节所述，更好的初始化以及激进但稳定的参数更新能够提升优化效率。接下来，我们将介绍如何将这一加速原则应用于 2DGS。

覆盖感知初始化。 我们首先通过覆盖感知初始化（CI）改善初始高斯分布。受 Image-GS [40], 启发，我们从由模糊图像梯度图导出的分布中采样高斯中心 $\boldsymbol{\mu}$，其中包含均匀采样成分。这样可以在细节丰富的区域放置更多高斯，同时保留无纹理区域的覆盖。我们将颜色 $\mathbf{c}'$ 初始化为零。对于形状，我们从 $[0,1]$ 中采样 ℓ_1，并从 $[0.5,1]$ 中采样 ℓ_0，ℓ_2，以避免高斯过小或过于细长，并改善初始覆盖。为了使高斯的初始空间范围适应不同的分辨率和高斯预算，我们将覆盖尺度定义为 $s_{\mathrm{cov}} = \min(H,\ W)^2/(\alpha N_t)$，其中 H 和 W分别表示帧高和帧宽，N_t表示高斯预算，α表示平衡系数。我们将初始 Cholesky 向量缩放为 s_{cov}，以调整高斯的空间范围。对于宽高比较高的图像，我们使用较短的图像维度，以防止高斯覆盖范围过大并产生重叠。对于给定分辨率，较小的高斯预算使用更大的覆盖范围，以改善初始覆盖并减少高斯增长所需的迭代次数；而较大的高斯预算使用更小的覆盖范围，以避免过度重叠及其导致的计算成本增加。如图 3 所示，CI 能够改善初始拟合期间的空间覆盖，这一点可以从 250 次迭代后的重建结果中看出。

激进且稳定的优化。 降低优化步数的一种直接方法是采用更大的更新幅度，但步长不能任意增大，否则会导致优化不稳定 [1]。如第 1 节所述，迭代式 2DGS 优化本身已经表现出明显的梯度波动和可见的闪烁。为支持激进的在线更新，我们引入基于指数移动平均的参数平均（EPA），使用指数移动平均（EMA）维护高斯参数的平滑副本 [28]。EPA 使在线参数能够进行激进更新，同时在梯度更新循环之外维护一条独立的指数平均参数轨迹。具体而言，我们维护两组高斯参数：在线参数 $\boldsymbol{\theta}^{(k)}$，其



(a) 不使用量化阶段覆盖感知初始化　(b) 使用量化阶段覆盖感知初始化

图 5：采用和未采用量化阶段覆盖感知初始化进行量化感知微调后的高斯结构。

通过激进的更新进行优化；以及指数移动平均参数 $\bar{\boldsymbol{\theta}}^{(k)}$，其在梯度更新循环之外进行维护。指数移动平均参数根据历史在线参数更新，公式如下：

$$\bar{\boldsymbol{\theta}}^{(k)} = \beta\bar{\boldsymbol{\theta}}^{(k-1)} + (1-\beta)\boldsymbol{\theta}^{(k)}, \tag{5}$$

其中 $\beta \in [0,1)$ 控制平均强度。展开公式 (5) 可得

$$\bar{\boldsymbol{\theta}}^{(k)} = (1-\beta)\sum_{j=1}^{k}\beta^{k-j}\boldsymbol{\theta}^{(j)} + \beta^k\bar{\boldsymbol{\theta}}^{(0)}. \tag{6}$$

因此，指数移动平均参数沿优化轨迹形成指数加权平均，在保留长期优化趋势的同时平滑短期参数波动。我们定期使用指数移动平均参数评估损失以进行收敛检测，并在最终在线参数与性能最佳的指数移动平均参数之间保留表现更好的状态。由于 $\bar{\boldsymbol{\theta}}$ 不参与梯度计算，EPA 不会改变 $\boldsymbol{\theta}$ 的在线更新。鉴于图 1b 所示 2DGS 中存在显著的梯度波动，我们采用较大的 EMA 衰减因子来平滑参数波动。指数移动平均参数无需计算梯度；图 1b 中显示的梯度是经过 EMA 平滑的在线梯度，仅用于可视化。

3.3 Fast-2DGS 视频编码

高斯视频编码。 我们首先介绍 GSVC 中的编码流程，这是整个方法的骨干。在表示层，帧被分为锚帧和传播帧，分别类似于帧内编码帧和帧间编码帧。在单遍模式下，锚帧使用新初始化的帧内高斯从头训练一次，而传播帧则通过连续学习继承并优化前一帧中的帧间高斯。在双遍模式下，GSVC 首先优化锚帧及其后的 k个传播帧，然后使用优化后的第 k帧重新初始化锚帧，并进一步对其进行优化，之后再次向前传播。最终只存储锚帧，相较于单遍编码不会增加存储开销。为适应视频动态，高斯泼溅增强（GSA）添加 $\rho_{\mathrm{dyn}}N_t$ 个新的帧内高斯，而高斯泼溅剪枝（GSP）则每经过 ΔK_r 次迭代，逐步从 K_r^{start}到 K_r^{end}移除相同数量的低贡献高斯。在检测到场景变化时，动态关键帧选择（DKS）会选择新的锚帧。在编码层，帧被编码为 I帧或 P帧。锚帧被编码为 I帧，其中所有高斯均独立编码。传播帧可以被编码为 P帧：新添加的高斯采用帧内编码，而继承的帧间高斯则作为残差进行编码

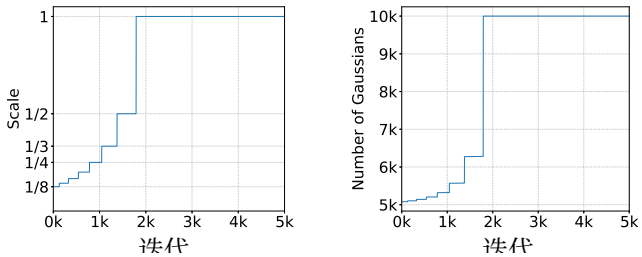
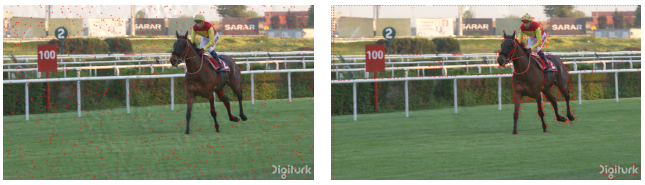


图 6：展示迭代过程中分辨率尺度（左）和高斯函数数量（右）的渐进式调度示例。



(a) 随机 **(b) 光流引导**

图 7：采用不同高斯注入策略初始化传播帧。红点表示新添加的帧内高斯的中心。

从而使用更少的比特。为限制图像组内的量化误差传播，可以将选定的传播帧改为帧内编码为 I帧，此时所有高斯均独立编码，而不是作为帧间残差进行编码。

初始化。CI 在两个优化阶段针对不同组件：表示优化阶段的高斯参数，以及量化感知微调阶段的量化器。在表示阶段，CI 会初始化新添加的帧内高斯，而帧间高斯则继承自前一帧。为消除对随机种子的依赖，我们使用固定且未经扰乱的 Sobol序列 [25]，替代随机采样，包括梯度引导初始化中的均匀成分。在量化阶段，高斯已经完成优化，因此 CI 转而初始化量化器。高斯形状参数以 Cholesky 系数 ℓ 表示，并使用 LSQ+ [3] 进行量化：$\hat{\ell} = s \cdot Q_b\left(\frac{\ell - b}{s}\right) + b$，其中 Q_b 表示 b 位的舍入与裁剪，s 和 b 分别表示可学习的量化尺度和偏移量。GSVC 使用较小的值初始化 s 和 b 以减少舍入误差，但这会缩小量化后的高斯，并在微调后破坏其学习到的形状，如图 4a 和图 5a 所示。相反，我们初始化 $s = 1$ 和 $b = 0$，以在微调前保留高斯形状，并在微调后保留其空间结构，如图 4b 和图 5b 所示。

优化。我们将 EPA 应用于表示优化和量化感知微调。由于训练过程中可能动态添加或移除高斯，因此会相应更新 EMA 参数集，使其始终与在线模型保持一致。EPA 在预热 K_{w} 次迭代后启动，以避免早期不稳定的参数污染 EMA 轨迹。我们还针对不同阶段调整学习率：锚帧采用较大的学习率η_{A}，以加快从头重建；传播帧和量化感知微调则采用较低的学习率 η_{P} 和η_{Q}，以保留继承而来或已优化的结构。

3.4 由粗到细重建

为进一步提升锚帧从头训练时的训练效果，我们在 CI 的基础上引入由粗到细渐进训练（PT）。我们不同时对所有高斯进行优化，使其逼近全分辨率目标，而是先建立粗略结构，再逐步细化细节。受 Dash-Gaussian [5]，启发，我们通过同时提高帧分辨率和高斯预算，逐步增加重建复杂度。

我们首先构建多分辨率重建目标。遵循 DashGaussian [5]，，我们在傅里叶域进行抗混叠下采样。给定一个锚帧 $\mathbf{F}_t$ 及其居中的傅里叶频谱 $\mathbf{Z}_t$，下采样因子为 r时的目标通过保留大小为 $H/r \times W/r$的中心频率区域 $\Omega(r)$并应用傅里叶逆变换得到。随着 r逐渐减小至 1，更高频率的细节会逐步引入。

我们不采用固定分辨率调度，而是根据保留的傅里叶幅值之和调整下采样因子 r，其定义如下：

$$\chi_t(r) = \sum_{\omega \in \Omega(r)} |\mathbf{Z}_t(\omega)| \,. \tag{7}$$

我们首先通过二分搜索求解 $\chi_t(r_{\max}) = \chi_t(1)/4$，以确定最粗下采样因子 $r_{\max}$。这样可以根据每个锚帧的频谱分布调整起始分辨率。然后，我们构建从 $r_{\max}$ 到 $r = 1$ (全分辨率)的下采样调度。具体而言，我们在 $\chi_{\min} = \chi_t(r_{\max})$ 与 $\chi_{\max} = \chi_t(1)$ 之间均匀采样 m 个显著性级别 $\{\chi_i\}_{i=0}^{m-1}$，并通过二分搜索求解对应的下采样因子 r_i，使其满足 $\chi_t(r_i) = \chi_i$。每个显著性级别都映射到一个过渡迭代：

$$s_i = K_{\mathrm{w}} \frac{\log(\chi_i/\chi_{\min})}{\log(\chi_{\max}/\chi_{\min})}, \tag{8}$$

其中，在 EMA 预热迭代 K_{w} 时达到全分辨率。在相邻过渡点 $(r_i,\ s_i)$ 和 $(r_{i+1},\ s_{i+1})$ 之间，我们遵循 DashGaussian [5]，并对下采样因子的平方倒数进行插值：

$$r_k = \left[(1-\lambda_k)\frac{1}{r_i^2} + \lambda_k \frac{1}{r_{i+1}^2}\right]^{-\frac{1}{2}}, \qquad \lambda_k = \frac{k - s_i}{s_{i+1} - s_i}. \tag{9}$$

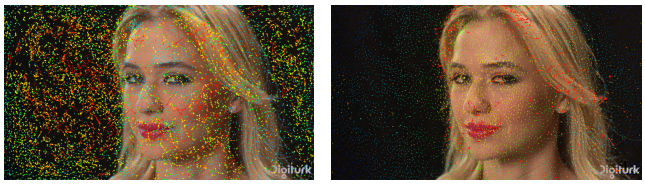
如图 6 所示，对数转换映射、插值和舍入共同作用，使较低分辨率分配到更少的迭代次数，并随着分辨率提高而逐步增加迭代次数。最后，我们将高斯预算与分辨率递进过程相结合。在迭代 k 时，高斯数量为

$$N_k = N_0 + \frac{N_t - N_0}{r_k^{2-k/K_{\mathrm{w}}}}, \qquad N_0 = \frac{N_t}{2}. \tag{10}$$

随着分辨率提高，高斯预算逐渐增加至 N_t，以捕捉更精细的细节。

3.5 光流引导传播

对于传播帧中的连续学习，我们使用光流来更好地利用时间一致性并加速训练。



（a）无时空约束　　(b) 使用时空约束

图 8：量化微调后的高斯间运动。

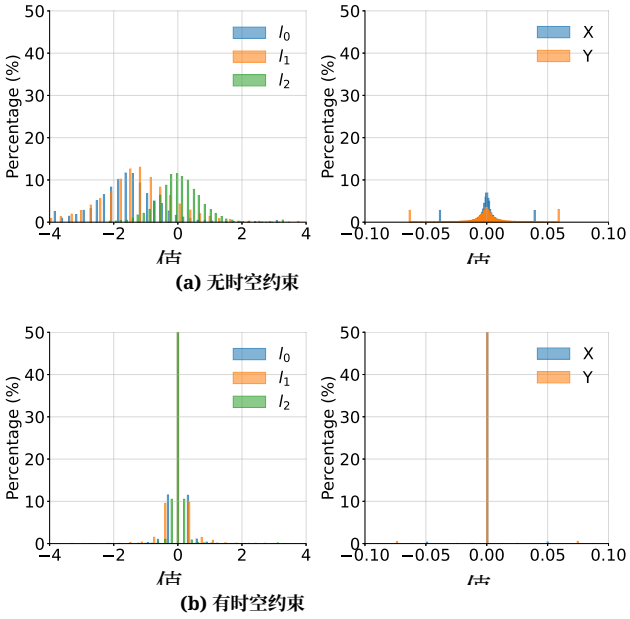


图9：量化微调后帧间高斯参数残差的分布。

光流引导初始化。 在训练之前，我们添加 $\rho_{\text{dyn}}N_t$ 新的帧内高斯。我们使用光流引导来定位扭曲误差较大的区域，如图 7 所示。具体而言，对于传播帧 $\mathbf{F}_t$ 及其前一帧 $\mathbf{F}_{t-1}$，我们计算反向光流 $\mathbf{U}_{t\to t-1}$。对于 $\mathbf{F}_t$中的每个像素 $\mathbf{p}$，该光流将其映射到 $\mathbf{F}_{t-1}$中的对应源位置：

$$\mathbf{q} = \mathbf{p} + \mathbf{U}_{t\to t-1}(\mathbf{p}). \tag{11}$$

˜我们在$\mathbf{F}_{t-1}$处采样$\mathbf{q}$，以构造扭曲后的前一帧$\mathbf{F}_{t-1\to t}$，并将逐像素扭曲误差计算为

$$e_t(\mathbf{p}) = \left\|\mathbf{F}_t(\mathbf{p}) - \tilde{\mathbf{F}}_{t-1\to t}(\mathbf{p})\right\|_1. \tag{12}$$

这样一来，新出现的内容、显露区域以及不准确的运动对应关系通常会产生较大的扭曲误差，表明这些区域难以通过时间传播进行解释，因而可能受益于新增高斯。为避免在高度动态区域出现过度重叠，我们将误差图划分为与图块同尺寸的单元，并在排名最高的各单元中，将一个高斯放置在误差最大的像素处。其余高斯则随机放置。由于光流仅作为粗略引导，其误差可由新引入的高斯加以纠正，因此我们主要根据流估计器的轻量化设计来选择它，详见第4节。

光流引导优化。 我们进一步将光流用作高斯间优化的运动先验。对于每个高斯间 i，我们在其投影中心以及沿水平和垂直方向偏移半径一半的位置处采样前向光流，并对有效采样值取平均，以获得其运动目标：

$$\bar{\mathbf{u}}_i = \frac{1}{|\mathcal{V}_i|}\sum_{\mathbf{q}\in\mathcal{V}_i}\mathbf{U}_{t-1\to t}(\mathbf{q}), \tag{13}$$

其中，$\mathcal{V}_i$ 表示帧间高斯 i的有效采样位置。在优化过程中，我们将高斯光流作为运动先验：

$$\mathcal{L}_{\text{flow}} = \frac{1}{N_{\text{inter}}}\sum_{i=1}^{N_{\text{inter}}}\left\|(\mathbf{p}_i^{(k)} - \mathbf{p}_i^{(0)}) - \bar{\mathbf{u}}_i\right\|_2^2, \tag{14}$$

其中，$\mathbf{p}_i^{(0)}$ 和$\mathbf{p}_i^{(k)}$ 分别表示帧间高斯 i的初始位置和当前位置。运动先验与重建损失共同进行优化：

$$\mathcal{L}^{(k)} = \mathcal{L}_{\text{rec}} + \lambda_{\text{flow}}^{(k)}\mathcal{L}_{\text{flow}}, \qquad \lambda_{\text{flow}}^{(k)} = \begin{cases}\lambda_{\text{flow}}, & k < K_{\text{flow}},\\ 0, & k \geq K_{\text{flow}},\end{cases} \tag{15}$$

其中 $K_{\text{flow}} = \min\left(K^{\text{conv}}{}_{\text{光流}}, K_r^{\text{start}}\right)$，$K^{\text{conv}}{}_{\text{光流}}$ 表示光流先验的收敛迭代，而 K_r^{start}表示高斯移除。在早期优化期间，L_{flow} 将继承的帧间高斯引导至其估计运动，从而减少对其新位置的搜索。一旦光流先验收敛或高斯移除开始（以先发生者为准），优化便仅依靠L_{rec}继续进行。新添加的帧内高斯不受运动先验影响，因为L_{flow} 仅定义于帧间高斯。

3.6 时序感知量化微调

对于压缩，我们通过时间感知量化（TAQ）利用时间冗余，为帧内高斯和经时间传播的帧间高斯采用不同的精度。具体而言，对于独立编码的帧内高斯，我们对位置使用16位精度，对颜色使用两阶段8位RVQ [7, 36]，对Cholesky参数使用6位LSQ + [3]。对于帧间高斯，我们编码其残差。时间一致性使这些残差集中在零附近，从而能够采用更低精度的量化，同时保持重建质量：对位置使用8位LSQ+，对颜色使用两阶段7位RVQ，对Cholesky参数使用5位LSQ+。码本通过承诺损失 [39],进行优化，并使用Linde–Buzo–Gray（LBG）聚类 [14], 进行确定性构建，以替代GSVC [31]所采用的随机K-means初始化。有关量化器公式的详细信息，请参阅GSVC [31]。

由于相较于全精度表示，低比特量化会引入精度损失，因此需要进行量化感知微调以恢复重建质量。然而，在补偿量化误差时，帧间高斯可能会偏离其参考高斯，从而削弱先前建立的时间一致性。如图8a所示，高斯运动变得

算法1 阶段I：表示优化

需要： 视频帧 $\mathrm{F} = \{\mathrm{F}_t\}_{t=1}^{T}$，高斯预算 $\{N_t\}_{t=1}^{T}$。
确保： 优化后的高斯 $\{\mathrm{G}_t^*\}_{t=1}^{T}$。

如果 $T > 1$ **则**
  使用 DKS 选择锚帧。
结束 if
对 $t = 1$ **到** T **执行 do**
  如果 F_t 是锚帧 **则**
    使用 CI 初始化 G_t；根据
    方程（9）和（10）计算渐进式分辨率和高斯调度。
  else
    从 G_{t-1} 继承帧间高斯；添加 $\rho_{\mathrm{dyn}} N_t$ 个帧内高斯
    使用方程（12）在基于流的高误差区域中加入 CI。
  结束 if
  当 未收敛时，$k = 0$，1，2，.....**执行**
    如果 F_t 是锚帧**则**
      更新渐进式调度，并使用 L_2 损失优化 G_t。
    else
      使用方程（15）通过运动先验优化 G_t。
    结束 if
    如果 $K_r^{\mathrm{start}} \le k \le K_r^{\mathrm{end}}$**且** $(k - K_r^{\mathrm{start}})$ 取模 $\Delta K_r = 0$ **则**
      移除贡献较低的高斯。
    结束 if
    如果 $k \ge K_w$ **则**
      使用公式（5）中的 EPA 初始化/更新 $\bar{\mathrm{G}}_t$。
    结束 if
    if 达到收敛检查间隔**then**
      使用指数移动平均参数评估损失。
      更新收敛状态和最佳指数移动平均检查点。
    结束 if
  结束 while
  在最终的 G_t^{last} 和最佳 $\bar{\mathrm{G}}_t$ 之间选择最佳模型，记为 G_t。
  如果 F_t 是锚帧**且**采用双遍编码**则**
    将 G_t 传播 through the next k frames 以获得 G_{t+k}。
    使用 G_{t+k} 重新初始化锚帧，并重新优化以获得 $\bar{\mathrm{G}}_t$。
  结束如果
结束循环

更加不规则，导致残差更加分散。为在量化感知微调过程中保持时间一致性，我们引入时空约束（STC），对帧间高斯的位置残差和形状残差进行正则化。对于帧 t 中的帧间高斯 n，我们分别用 $\Delta\boldsymbol{\mu}_n^t$ 和 $\Delta\boldsymbol{I}_n^t$表示其相对于参考高斯的位置残差和 Cholesky 残差。随后，我们优化

$$\mathcal{L}_{\mathrm{q}} = \mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{q}} \frac{1}{|\mathcal{N}_t^{\mathrm{Inter}}|} \sum_{n \in \mathcal{N}_t^{\mathrm{Inter}}} \left(\eta_t \left\| \Delta\boldsymbol{\mu}_n^t \right\|_1 + \left\| \Delta\boldsymbol{I}_n^t \right\|_1 \right), \tag{16}$$

其中，$\mathrm{N}_t^{\mathrm{Inter}}$表示帧 t 中帧间高斯的索引集合。由于重建对高斯位置更加敏感，我们对第一个 P 帧中的帧间高斯位置不加约束，使其能够吸收从锚帧传播而来的量化误差。因此，从第二个 P 帧开始施加位置约束（第一个 P 帧使用 $\eta_t = 0$，其余帧使用 1），而所有 P 帧中的 Cholesky 残差均受到约束。如图 8b 和 9 所示，这种正则化有助于高斯运动更加连贯，并使残差分布更加集中。

算法 2阶段 II：量化感知微调

需要： 视频帧 $\mathrm{F} = \{\mathrm{F}_t\}_{t=1}^{T}$、优化后的高斯 $\{\mathrm{G}_t^*\}_{t=1}^{T}$、锚帧列表、GOP 设置。
  锚帧列表、GOP 设置。
确保： 量化后的高斯 $\{\tilde{\mathrm{G}}_t\}_{t=1}^{T}$。

对于 $t = 1$ **到** T **执行**
  使用 CI 初始化量化器
  如果 F_t 是一个 I帧，**则**
    将所有高斯量化为帧内高斯。
  else
    相对于 $\hat{\mathrm{G}}_{t-1}$ 量化帧内高斯和帧间高斯残差。
  结束如果
  当 尚未收敛时，$k = 0$，1，2，.....**执行**
    量化并反量化 G_t。
    如果F_t 是一个 I帧 **则**
      使用重建 L_2 损失优化 G_t。
    else
      相对于 $\hat{\mathrm{G}}_{t-1}$ 恢复 G_t，并使用公式(16)进行优化。
    结束如果
    如果 $k \ge K_w$ **则**
      使用公式（5）中的 EPA 初始化/更新 $\bar{\mathrm{G}}_t$。
    结束 if
    如果 达到收敛检查间隔**则**
      使用指数移动平均参数评估损失。
      更新收敛状态和最佳指数移动平均检查点。
    结束 if
  结束 while
  在最终 G 中选择最佳模型 G_t^{last} 和最佳 $\bar{\mathrm{G}}_t$，并将其量化为输出 $\tilde{\mathrm{G}}_t$；反量化为 $\hat{\mathrm{G}}_t$，用于后续 P 帧。
结束循环

3.7 编码流程与执行

总的来说，Fast-GSVC 流程包括算法1所示的表示优化、算法 2 所示的量化感知微调，以及熵编码。前两个阶段都采用 CI 和 EPA。表示优化还对锚帧采用渐进式训练，并对传播帧采用光流引导；量化感知微调则采用时序感知量化和时空约束。为减少运行时调度开销，我们采用 CUDA 图 [19]，以更少的 CPU启动重放重复的图形处理器操作。在表示优化期间，图像分辨率和高斯数量固定后便开始重放图。在量化感知微调期间，模型结构和高斯数量保持固定，因此初始化后即可捕获重复的优化步骤。由于 CUDA 图只改变执行调度，而保留底层优化操作，因此可以降低内核启动开销，同时不影响重建质量。最后，所有量化参数均直接进行熵编码，但 16 位位置参数除外；其高、低 8 位字节分别编码。所有参数的符号范围均采用硬编码，以进一步降低熵编码开销。

4 实验

4.1 实验设置

数据集和指标UG 数据集中的所有 1080p 视频上评估 Fast-GSVC [17],，这些视频以 1920×1080 分辨率、120 fps 和 8-bit YUV 4:2:0格式提供。遵循 GSVC [31],，我们使用

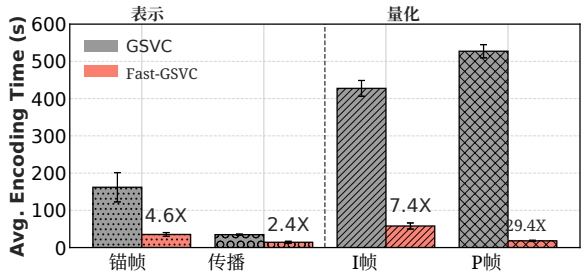


图10：GSVC 和 Fast-GSVC 在锚帧、传播帧、I帧和 P帧处理阶段的平均编码时间。误差棒表示 ± 1 标准差。

每个视频的前 50 帧。我们报告 PSNR、多尺度结构相似性 [33], 和视频多方法评估融合 [2] 以衡量重建质量，使用每像素比特数衡量压缩效率，并报告端到端实际编码时间，其中包括高斯初始化、光流处理、表示优化、量化感知微调和熵编码。较高的 PSNR、多尺度结构相似性和视频多方法评估融合表示更好的重建质量，而较低的每像素比特数和编码时间分别对应更小的压缩文件大小和更快的编码速度。

实现。我们采用 IPPPP GOP结构，高斯预算为 {9k, 18k, 27k, 36k, 45k}。每 100 次迭代通过 EMA重建损失检查收敛情况；若连续 10 次检查中损失改善幅度均小于 1%，则判定为收敛。对于 CI，平衡系数 α 设为 32，高斯中心通过 90% 的梯度引导采样和 10% 的均匀采样混合方式进行采样。对于 EPA，我们采用 $K_w = 2500$ 次迭代的预热和 $\beta = 0.99$的 EMA 衰减因子，这与常见的 EMA 平滑做法一致 [18]。为确保预热稳定，锚帧学习率初始化为 10^{-3}，随后 K_w，之后提高至 10^{-2}；传播帧学习率则固定为 10^{-3}。双遍编码设为 $k = 1$。对于渐进式训练，我们使用 $m = 32$ 个下采样层级。我们额外分配 $\rho_{\text{dyn}} = 10\%$ 的动态高斯预算，并每隔 $\Delta K_r = 100$ 次迭代移除高斯：锚帧从 $K_r^{\text{start}} = 0$ 到 $K_r^{\text{end}} = 4000$，传播帧从 $K_r^{\text{start}} = 500$ 到 $K_r^{\text{end}} = 1000$。光流先验的权重为 $\lambda_{\text{flow}} = 10^{-3}$。在量化感知微调期间，我们采用 10^{-3} 的学习率，并设定 $\lambda_{\text{q}} = 10^{-8}$。对于 I帧，收敛后进一步将学习率减半两次以进行细化。同时，我们将量化间隔的下限设为其初始值的 5%，以避免过度码值截断。由于 YachtRide 和 Jockey 的运动更复杂，其熵友好损失权重设为 10^{-5}；其他序列则设为 10^{-4}。由于我们只需要粗略的运动先验，且光流误差可通过新添加的高斯进一步校正，因此使用轻量级、基于 CPU 的 OpenCV DIS估计器，以半分辨率计算双向光流，耗时 ~102 毫秒。所有实验均在 NVIDIA H100 GPU 上进行。

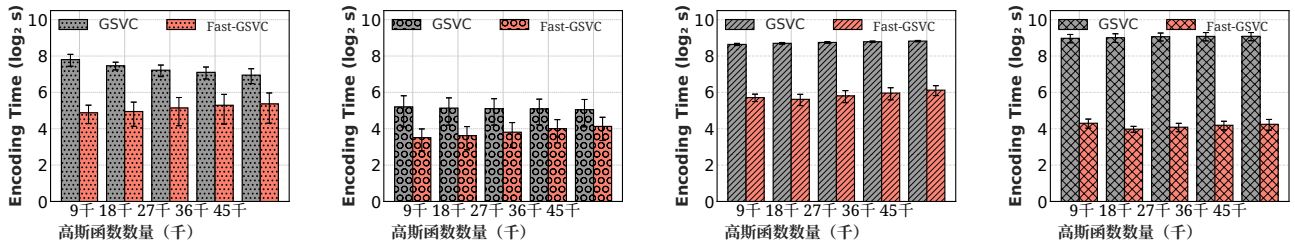
对比方法。我们首先将 GSVC [31], 这一当前最先进的迭代式二维高斯视频编解码器作为对比方法，并采用相同的高斯预算和 GOP结构（IPPPP）。由于两种方法采用不同的优化过程，我们不强制要求迭代次数相同。相反，我们测量各方法在其原始优化设置下达到最佳性能所需的编码时间。因此，我们首先比较编码时间，然后评估相应编码时间下的率失真性能，以证明 Fast-GSVC 实现了第 3.1 节提出的目标：以更短的编码时间获得更好的率失真性能。此外，我们还报告我们的方法（*IP~*），其中仅将第一帧编码为 I帧，其余所有帧均编码为 P 帧，以评估长期时间传播效果。我们还将标准化视频编解码器 HEVC [26] 和 VVC [4], 作为对比，分别使用 HM-16.2 和 VTM-23.6。两者均采用相同的 IPPPP GOP结构，QP值范围为 18 至 28。此外，我们还纳入当前最先进的前馈式2DGS方法 AIR [38]，以考察前馈预测作为一种替代方案。

4.2 结果与分析

编码加速。我们首先报告双遍编码的锚帧、传播帧、I帧和P帧的编码时间，这些时间均在所有测试的高斯预算设置下取平均值。如图10所示，Fast-GSVC 显著加快了编码速度。在表示优化阶段，双遍编码后，锚帧实现了 4.6× 倍加速比，传播帧实现了 2.4× 倍加速比。在量化感知微调阶段，I 帧和 P 帧的加速比分别进一步达到 7.4× 倍和 29.4× 倍。值得注意的是，主导编码瓶颈从 GSVC 中的 P 帧转移到了 Fast-GSVC 中的 I 帧，这表明所提出的 P 帧设计能够更有效地利用时间一致性。对于采用 IPPPP 编码结构的 50 帧视频，Fast-GSVC 将总处理时间从约 7.56 小时缩短至 34 分钟，实现了 13.4× 倍的总体加速比，并将编码时间减少了 92.5%。

为进行更详细的比较，我们在图 11 中进一步报告了每种高斯预算下的编码时间。Fast-GSVC 在所有测试的高斯预算下都能持续加速编码。具体而言，双遍编码后，锚帧的加速比范围为 3.1–8.0×，传播帧为 2.0–3.3×，I 帧为 6.5–7.7×，P 帧为 25.9–31.0×。有趣的是，对于 GSVC 的锚帧优化，编码时间会随着高斯预算的增加而减少，而 Fast-GSVC 的编码时间则会随预算增加而增加。这是因为，在较小预算下，对于 GSVC 而言，随机初始化的高斯无法提供充分的图像覆盖，并且如第 1 节所述，需要更多早期优化步骤来建立足够的覆盖。相比之下，Fast-GSVC 中的覆盖感知初始化在开始时即可提供充分的覆盖，从而消除了这一早期优化低效问题。

率失真权衡。随后我们展示了，借助上文所报告的大幅缩短的编码时间，Fast-GSVC 仍能实现更优的率失真性能。图 12 展示了总体率失真性能。在整合为完整流程后，Fast-GSVC 在三项指标上均达到与 GSVC 相当（VMAF）或更优（PSNR、多尺度结构相似性）的性能。在相同 PSNR 下，Fast-GSVC 可将每像素比特数降低约 46%（38–51%）；在相同每像素比特数下，PSNR 可提升约 1.9 分贝（1.5–2.2 分贝）。采用*Ours (IP~)*后，收益进一步提升至每像素比特数降低 58%（51–63%），PSNR 提升 2.5 分贝（2.2–2.8 分贝）。尽管*Ours (IP~)*在相同高斯预算下的量化后质量略低，但其更长的 P 帧链能够更充分地利用时间冗余，从而产生熵更低的残差和更小的比特流。这进一步证明了时间预测



(a) 锚帧 (b) 传播帧 （c）I帧 （d）P帧

图 11：编码时间与高斯预算的对数$\log_2$尺度关系：表示优化（a、b）和量化（c、d）。误差线表示 ± 1 标准差。

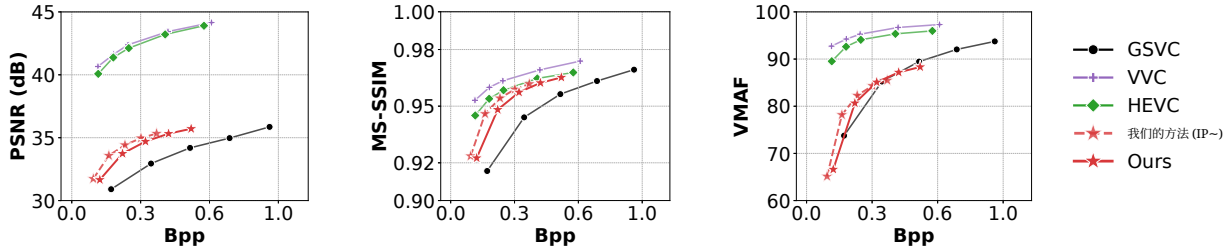


图 12：Fast-GSVC 与 UVG 数据集上各基线方法的量化率失真结果。

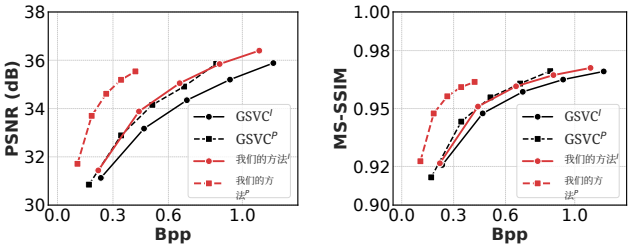


图 13：GSVC 与 Fast-GSVC 在 I 帧和 P 帧上的率失真对比。

和 Fast-GSVC 中残差编码的有效性。此外，如图 13 所示，Fast-GSVC 在 I 帧和 P 帧上均持续优于 GSVC。GSVC 运行 100k 次迭代并选取其最佳检查点，而 Fast-GSVC 在 I 帧和 P 帧上平均仅需分别进行 23.7k 次和 6.2k 次迭代，这表明只需大幅减少优化迭代次数，也能实现更优的率失真性能。为完整起见，我们还与标准化视频编解码器进行了比较。如图 12 所示，尽管 Fast-GSVC 在 PSNR 和 VMAF 上仍落后于 HEVC 和 VVC，但在更高比特率下，其多尺度结构相似性达到相当水平。更重要的是，与 GSVC 相比，Fast-GSVC 在大幅缩短编码时间的同时，缩小了 PSNR 和多尺度结构相似性方面与这些标准化编解码器之间的率失真差距。目前，我们采用 L_2 重建项，以降低计算成本。引入感知损失可能会改善视觉质量，但也会增加编码时间，这一点已在动态高斯重建 [35]中观察到。更严格的收敛准则和额外的迭代次数可以进一步提升峰值信噪比，但代价是编码时间增加 [31]。除率失真性能外，显式的二维高斯表示还能够直接控制各个基元，为自适应高斯分配、细化和渐进式编码提供了灵活性 [32]。

消融研究。 我们从加速和压缩性能两个层面开展消融研究。对于加速效果，我们在三种设置下评估速度—质量权衡：从头训练（单遍锚帧优化）、微调（量化）和连续优化（传播帧优化）。图 14 展示了实验结果，其中 无 表示不包含加速组件的基线。每个点对应五种高斯预算中的一种，并报告相同预算下相对于无 的峰值信噪比变化。如图 14a 和图 14b 所示，所有加速变体都能缩短训练时间，同时逐步加入 EPA、PT 和 CI 可提升重建质量，验证了它们在实现更佳速度—质量权衡方面的有效性。由于光流引导专用于传播帧优化，我们在图 14c 中对其单独进行消融；结果表明，它能够进一步改善速度—质量权衡。对于压缩性能，我们在图 15 中对时序感知量化（TAQ）和时空约束（STC）进行了消融。与基线相比，TAQ 提升了率失真性能，而 STC 进一步带来增益，验证了这两个组件的有效性。

前馈初始化。 前馈方法无需逐实例优化即可实现快速重建，但这类方法不太适用于视频编码，因为视频编码需要能够精细控制率质量的“旋钮”。针对每段视频进行优化的

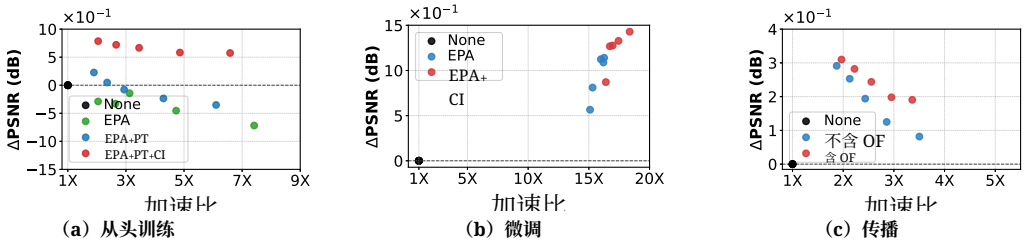


图14：在三种优化设置下，对加速组件在速度–质量权衡方面的消融实验：(a) 从头训练，(b) 量化微调，以及 (c) 传播帧优化。

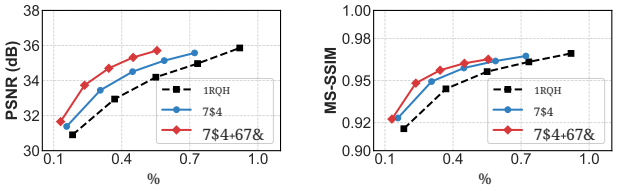


图15：时间感知量化（TAQ）和时空约束（STC）的率失真消融实验。

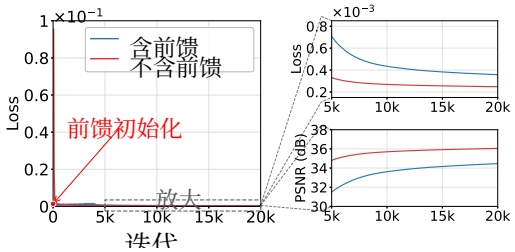


图 16：采用与不采用前馈初始化的 Fast-GSVC 的优化动态。

2DGS 视频编解码器天然支持以任意高斯预算进行恒定码率编码，也支持通过渐进式剪枝高斯以达到目标质量 [31] 来进行恒定质量编码。相比之下，前馈方法中每像素生成的高斯函数数量与学习网络紧密耦合，因此更改预算需要进行架构修改和重新训练。正如我们在下文实验中所观察到的，直接剪枝前馈预测结果也可能颇具挑战。

结合二者优势的一种可能方式是使用前馈预测进行初始化，并在后续优化过程中逐步剪枝预测得到的高斯。我们使用当前最先进水平的前馈式2DGS方法 AIR [38] 对这一策略进行了评估。具体而言，AIR 首先为 Jockey 的锚帧预测 10,764 个高斯，随后将其逐步剪枝至 9K 的目标预算；我们的初始化从相同数量的高斯开始，并采用相同的剪枝计划。如图16所示，前馈初始化以更低的损失提供了更好的起点。然而，AIR 生成的高斯初始化重叠程度高得多，平均每个像素由 8.3 个高斯覆盖，而 Fast-GSVC 仅为 3.9 个。在采用相同的渐进式剪枝计划后，AIR 初始化收敛到的解的峰值信噪比约低 2 分贝。这表明，在我们的实验中，AIR 初始化对剪枝更加敏感，而其根本原因仍需进一步研究。尽管 Fast-GSVC 目前尚未采用现成的前馈初始化，但联合设计前馈预测、高斯剪枝和后续优化仍是未来值得探索的方向。

计算成本。启用所有提升优化效率的加速组件后，GPU 内存峰值增加从 9千至 45千个高斯时的 824–838 MiB 增至 874–892 MiB，相应的额外开销仅约为 50 MiB。这一增幅主要来自在优化循环之外维护分离的指数移动平均参数；由于它们既不需要梯度，也不需要优化器状态，因此额外内存占用很小。启用 CUDA图后，训练时间进一步缩短 36%–46%（1.56×–1.85× 倍加速比），而 GPU 内存峰值则增至 1,278–1,338 MiB。额外内存主要用于回放所需的持久化图缓冲区。在我们的实验中，为缓冲区预留的容量是预热期间观测到的峰值内存需求的三倍，以适应运行时间变化，避免动态重新分配；更精确的容量估算可进一步降低这项开销。相比之下，AIR [38] 的前馈初始化使用最小的补丁大小为 14 的模型生成约 10千个高斯时，需要约 6.6 GB 的 GPU 内存。关于基于 2DGS 的方案与标准化视频编解码器之间的解码计算成本，请参阅扩展版 GSVC 研究 [31]，了解解码延迟和能源效率。

5 结论

在本文中，我们将缓慢的优化收敛确定为 2DGS 视频编码中的关键瓶颈。我们通过改进优化效率来解决这一瓶颈，采用了多项互补设计，包括更好的初始化、稳定且激进的参数更新、渐进式优化以及时间传播引导，同时通过 GPU 级优化进一步降低运行时间开销。此外，我们通过时序感知量化提升率失真性能，并通过时空约束保持时间一致性。综合而言，这些设计使 Fast-GSVC 相较于

GSVC 实现了 13.{v1} 倍编码加速，将平均编码时间从每帧约 9 分钟缩短至 40 秒，同时在重建质量相当的情况下，文件大小最多减少 58%。这项工作使二维高斯视频编码更快速、更紧凑且质量更高，缩小了其迈向实际部署的差距。我们还将正式的主观评估留待未来工作。尽管实现了加速，逐帧迭代优化仍然是实时编码的瓶颈。未来工作可以进一步探索前馈高斯预测，在保持压缩效率和可扩展性的同时降低优化成本。

参考文献

[1] Kwangjun Ahn、Jingzhao Zhang 和 Suvrit Sra。2022。理解梯度下降的不稳定收敛。载于《国际机器学习会议》（ICML 2022），2022年7月17日至23日，美国马里兰州巴尔的摩（《机器学习研究论文集》，第162卷），Kamalika Chaudhuri、Stefanie Jegelka、Le Song、Csaba Szepesvári、Gang Niu 和 Sivan Sabato（编）。PMLR，247–257。https: //proceedings.mlr.press/v162/ahn22a.html [2] Christos G Bampis、Zhi Li 和 Alan C Bovik。2019。用于全参考视频质量评估的时空特征整合与模型融合。《IEEE视频技术电路与系统汇刊》29，8（2019），2256–2270。[3] Yash Bhalgat、Jinwon Lee、Markus Nagel、Tijmen Blankevoort 和 Nojun Kwak。2020。LSQ+：通过可学习偏移量和更好的初始化改进低比特量化。载于《2020年IEEE/CVF计算机视觉与模式识别会议》，CVPR 2020研讨会，2020年6月14日至19日，美国华盛顿州西雅图。计算机视觉基金会 / IEEE，2978 2985 10 1109 50498 2020 00356 –。doi：. / [4] CVPRW . . Benjamin Bross、Ye-Kui Wang、Yan Ye、Shan Liu、Jianle Chen、Gary J. Sullivan 和 Jens-Rainer Ohm。2021。《通用视频编码（VVC）标准及其应用概述》。《IEEE视频技术电路与系统汇刊》31，10，3736 3764 10 1109 2021 3101953 –。doi：. /TCSVT. .[5] Youyu Chen、Junjun Jiang、Kui Jiang、Xiao Tang、Zhihao Li、Xianming Liu 和 Yinyu Nie。2025。DashGaussian：在200秒内优化三维高斯泼溅。载于《IEEE/CVF计算机视觉与模式识别会议》，CVPR 2025，2025年6月11日至15日，美国田纳西州纳什维尔。计算机视觉基金会 / IEEE，11146–11155。doi:10.1109/CVPR52734.2025.01041[6] Zhenghao Chen、Zicong Chen、Lei Liu、Yiming Wu 和 Dong Xu。2025。采用生成式二维高斯泼溅的通用视频标记化。CoRR abs/2508.11183（ 2025 2508 11183 10 48550 2508 11183 ）。ar [7] Xiv: . doi: . /ARXIV. . Robert Gray。. 矢量量化。《IEEE Assp Magazine》1，2（1984），4–29。[8] Bernhard Kerbl、Georgios Kopanas、Thomas Leimkühler 和 George Drettakis。2023。用于实时辐射场渲染的三维高斯泼溅。《ACM图形学汇刊》。42, 4 2023 139 1 139 14 10 1145 3592433 ( )，：–：. doi：. / Inseo Lee、 [9] Youngyoon Choi 和 Joonseok Lee。. GaussianVideo：通过高斯泼溅实现高效视频表示与压缩。载于《IEEE/CVF计算机视觉与模式识别会议研讨会》，CVPR 2025研讨会，2025年6月11日至15日，美国田纳西州纳什维尔。计算机视觉基金会 / IEEE，4471–4480。https://openaccess.thecvf.com/content/CVPR2025W/PBVS/html/ Lee_GaussianVideo_Efficient_Video_Representation_and_Compression_by_Gaussian_Splatting_CVPRW_2025_paper.html [10] Jiahao Li、Bin Li 和 Yan Lu。2022。用于神经视频压缩的混合时空熵建模。载于《MM '22：第三十届ACM国际多媒体会议》，2022年10月10日至14日，葡萄牙里斯本。ACM，1503–1511。doi:10.1145/3503161.3547845[11] Jiahao Li、Bin Li 和 Yan Lu。2023。具有多样化上下文的神经视频压缩。载于《IEEE/CVF计算机视觉与模式识别会议》，CVPR 2023，2023年6月17日至24日，加拿大不列颠哥伦比亚省温哥华。IEEE，22616–22626。doi:10.1109/CVPR52729.2023.02166[12] Jiahao Li、Bin Li 和 Yan Lu。2024。采用特征调制的神经视频压缩。载于《IEEE/CVF计算机视觉与模式识别会议》，CVPR 2024，2024年6月16日至22日，美国华盛顿州西雅图。IEEE，26099–26108。doi:10.1109/CVPR52733.2024.02466[13] Tiantian Li、Xinjie Zhang、Xingtong Ge、Tongda Xu、Dailan He、Jun Zhang 和 Yan Wang。2026。GaussianImage++：利用二维高斯泼溅增强图像表示与压缩。载于第四十届AAAI人工智能会议、第三十八届创新人工智能应用会议、第十六届人工智能教育进展研讨会暨AAAI 2026，2026年1月20日至27日，新加坡，Sven Koenig、Chad Jenkins 和 Matthew E. Taylor（编）。AAAI 出版社，6442 6449 10 1609 40 8 37572 –。doi：. [14] /AAAI.V I . Yoseph Linde 、 Andres Buzo 和 Robert M. Gray。. 矢量量化器设计算法。《IEEE通信汇刊》28，1（1980），84–95。doi:10.1109/TCOM. 1980.1094577[15] Yueh-Cheng Liu、Lukas Höllein、Matthias Nießner 和 Angela Dai。2025。Quick-Splat：通过学习式高斯初始化实现快速三维表面重建。载于《IEEE/CVF国际计算机视觉会议》，ICCV 2025，檀香山，夏威夷州，美国，2025年10月19日至25日。IEEE，27851–27861。doi:10.1109/ICCV51701.2025.02585[16] Saswat Subhajyoti Mallick、Rahul Goel、Bernhard Kerbl、Markus Steinberger、Francisco Vicente Carrasco 和 Fernando De l a Torre。2024。Taming 3DGS：有限资源下的高质量辐射场。载于《2024年亚洲计算机图形学国际会议论文集》，SA 2024，日本东京，2024年12月3日至6日，Takeo Igarashi、Ariel Shamir 和 Hao (Richard) Zhang（编）。ACM，2:1–2:11。doi:10.1145/3680528.3687694[17] Alexandre Mercat、Marko Viitanen 和 Jarno Vanne。2020。UVG 数据集：用于视频编解码器分析与开发的50/120fps 4K序列。载于《第11届ACM多媒体系统会议论文集》，MMSys 2020，土耳其伊斯坦布尔，2020年6月8日至11日，Laura Toni、Ali C. Begen、Özgü Alay 和 Christian Timmerer（编）。ACM，297 302 10 1145 3339825 3394937 –。doi: . / . Daniel Morales-Brotons、 [18] Thijs Vogels 和 Hadrien Hendrikx。2024。深度学习中的权重指数移动平均：动态与优势。《机器学习研究汇刊》2024（2024）。https://openreview.net/forum?id=2M9CUnYnBA [19] 英伟达公司。 [n. d.]。CUDA图。CUDA编程指南。https://docs.nvidia.com/cuda/cuda-programming-guide/04-special-topics/cuda-graphs.html 访问日期：2026年9月18日。[20] Shiwei Ren、Tianci Wen、Yongchun Fang 和 Biao Lu。2026。FastGS：在100秒内训练三维高斯泼溅。载于《IEEE/CVF国际计算机视觉与模式识别会议》，CVPR 2026，美国科罗拉多州丹佛，2026年6月3日至7日。计算机视觉基金会，26094–26103。https://openaccess.thecvf.com/content/CVPR2026/html/Ren_FastGS_ Training_3D_Gaussian_Splatting_in_100_Seconds_CVPR_2026_paper.html [21] Yuang Shi、Géraldine Morin、Simone Gasparini 和 Wei Tsang Ooi。2025。LapisGS：用于自适应流媒体的分层渐进式三维高斯泼溅。载于《国际三维视觉会议》，3DV 2025，新加坡，2025年3月25日至28日。IEEE，991–1000。doi:10.1109/3DV66043.2025.00096[22] Yuang Shi、Géraldine Morin、Simone Gasparini 和 Wei Tsang Ooi。2026。Sketch&Patch++：高效的结构感知三维高斯表示。CoRR abs/2601 05394 2026 2601 05394 10 48550 2601 05394 。（）。arXiv：。 [23] doi：。/ARXIV。。Yuang Shi、Ruoyu Zhao、Simone Gasparini、Géraldine Morin 和 Wei Tsang Ooi。2024。通过基于神经网络的表示实现体积视频压缩。载于《第16届沉浸式混合与虚拟环境系统国际研讨会论文集》，MMVE 2024，意大利巴里，2024年4月15日至18日。ACM，85–91。doi:10.1145/3652212.3652220[24] Weronika Smolak-Dyzewska、Dawid Malarz、Kornel Howil、Jan Kaczmarczyk、Marcin Mazur 和 Przemyslaw Spurek。2026。VeGaS：视频高斯泼溅。《信息科学》735 2026 123033 10 1016 2025 123033 （），。doi：。/J.INS.。。Ilya [25] M Sobol。。立方体中的点分布与积分近似求值。《苏联计算数学与数学物理》7（1967），86–112。[26] Gary J. Sullivan、Jens-Rainer Ohm、Woojin Han 和 Thomas Wiegand。2012。高效视频编码（HEVC）标准概述。《IEEE视频技术电路与系统汇刊》。22，12（2012），1649–1668。doi:10.1109/TCSVT.2012.2221191[27] Yang-Tian Sun、Yihua Huang、Lin Ma、Xiaoyang Lyu、Yan-Pei Cao 和 Xi- aojuan Qi。2024。泼溅视频：用于多样化处理的视频高斯表示。载于《神经信息处理系统进展37：2024年神经信息处理系统年度会议》，NeurIPS 2024，加拿大不列颠哥伦比亚省温哥华，2024年12月10日至15日，Amir Globersons、Lester Mackey、Danielle Belgrave、Angela Fan、Ulrich Paquet、Jakub M. Tomczak 和 Cheng Zhang（编）。http://papers.nips.cc/paper_files/paper/2024/hash/5a461bdff86cc07e976bb6c518810398-Abstract-Conference.html[28] Antti Tarvainen 和 Harri Valpola。2017。均值教师是更好的榜样：权重平均一致性目标改善半监督深度学习结果。载于《第5届国际学习表征会议》，ICLR 2017，法国土伦，2017年4月24日至26日，研讨会分会场论文集。OpenReview.net。https://openreview.net/forum?id=ry8u21rtl [29] Chengbo Wang、Guozheng Ma、Yifei Xue 和 Yizhen Lao。2025。通过 Group Training 实现更快更好的三维高斯泼溅。载于《IEEE/CVF国际计算机视觉会议》，ICCV 2025，美国夏威夷州檀香山，2025年10月19日至25日。IEEE，27968–27977。doi:10.1109/ICCV51701.2025.02596[30] Longan Wang、Yuang Shi 和 Wei Tsang Ooi。2025。GSVC：通过二维高斯泼溅实现高效的视频表示与压缩。载于《第35届数字音频与视频网络及操作系统支持研讨会论文集》，NOSSDAV 2025，南非斯泰伦博斯，2025年3月31日至4月4日。ACM，15 21 10 1145 3712678 3721876 –。doi：。/。Longan [31] Wang、Yuang Shi 和 Wei Tsang Ooi。2025。GSVC：通过二维高斯泼溅实现高效的视频表示与压缩。CoRR abs/2501.12060。arXiv:2501.12060 doi:10.48550/ARXIV.2501.12060[32] Longan Wang、Yuang Shi 和 Wei Tsang Ooi。2026。P-GSVC：用于可扩展图像与视频的分层渐进式二维高斯泼溅。载于《2026年ACM多媒体系统会议论文集》，MMSys 2026，中国香港特别行政区，4月 *4 8, 2026* 156 166 10 1145 3793853 3795755 –。ACM，–。doi：。/。Zhou Wang、[33] Eero P Simoncelli 和 Alan C Bovik。2003。用于图像质量评估的多尺度结构相似性。载于《第37届阿西洛马会议：信号、系统与计算机》，第2卷。IEEE，1398–1402。

[34] Thomas Wiegand、Gary J. Sullivan、Gisle Bjøntegaard 和 Ajay Luthra。2003。H.264/AVC 视频编码标准概述。*IEEE*视频技术电路与系统汇刊。13，7（2003），560–576。doi:10.1109/TCSVT.2003。815165[35] Yukun Xu、Keyang Ye、Tianjia Shao 和 Yanlin Weng。2025。用于动态人体建模的可动画化三维高斯。《计算机科学前沿》19，9（2025），199704。doi:10.1007/S11704-024-40497-5[36] Neil Zeghidour、Alejandro Luebs、Ahmed Omran、Jan Skoglund 和 Marco Tagliasacchi。2022。SoundStream：端到端神经音频编解码器。*IEEE/ACM* 音频、语音与语言处理汇刊。30（2022），495–507。doi:10. 1109/TASLP.2021.3129994[37] Huimin Zeng、Yue Bai 和 Yun Fu。2026。任意尺度三维高斯超分辨率。载于第四十届*AAAI*人工智能会议、第三十八届人工智能创新应用会议、第十六届人工智能教育进展研讨会，*AAAI 2026*，新加坡，*2026*年*1*月*20*日至*27*日，Sven Koenig、Chad J enkins 和 Matthew E. Taylor（编）。AAAI 出版社，12304 12312 10 1609 40 15 38222–。doi：。/AAAI.V I。[38] Zhaojie Zeng、Yuesong Wang、Yawei Luo 和 Tao Guan。2026。AIR：用于自监督前馈二维高斯泼溅。*CoRR* abs/2605.20820（2026）。arXiv:2605.20820 doi:10.48550/ARXIV. 2605.20820[39] Xinjie Zhang、Xingtong Ge、Tongda Xu、Dailan He、Yan Wang、Hongwei Qin、Guo Lu、Jing Geng 和 Jun Zhang。2024。GaussianImage：通过二维高斯泼溅实现 1000 FPS 图像表示与压缩。载于《计算机视觉——*ECCV 2024*》第 *4* 卷，*2024* 年，第 *18–29* 页，*15067*——第 *29* 届欧洲会议论文集，意大利米兰，*9* 月—*10* 月，卷 *IX*（计算机科学讲义，第 卷），Ales Leonardis、Elisa Ricci、Stefan Roth、Olga Russakovsky、Torsten Sattler 和 Gül Varol（编）。施普林格，327 345 10 1007 978 3 031 72673 6_18 – 。doi: 。/ - - - - [40] Yunxiang Zhang、Bingxuan Li、Alexandr Kuznetsov、Akshay Jindal、Stavros Diolatzis、Kenneth Chen、Anton Sochenov、Anton Kaplanyan 和 Qi Sun。2025。Image-GS：基于二维高斯的内容自适应图像表示。载于 计算机图形学与交互技术特别兴趣小组会议论文集，*SIGGRAPH 2025* 会议论文集，温哥华，不列颠哥伦比亚省，加拿大，*2025* 年 *8* 月 *10—14* 日，Ginger Alford、Hao (Richard) Zhang 和 Adriana Schulz（编）。ACM，102:1–102:11。doi:10.1145/3721238.3730596[41] Fatimah Zohra、Chen Zhao、Shuming Liu、Yahya Al Malallah 和 Bernard Ghanem。2026。HyperGS：快速且可泛化的高斯视频表示。*CoRR* abs/2607.11500（2026）。arXiv:2607.11500 doi:10.48550/ARXIV.2607.11500

视觉比较

我们比较所有七　在 UVG 序列中使用全帧视图以及相同 600×150 像素区域的 3.2× 放大视图　.

A.1 收敛速度

图 17 和图 18 对比了经过 2,000 次和 4,000 次迭代后的重建结果。在这两个检查点，Fast-GSVC 的峰值信噪比均始终高于 GSVC，表明其收敛速度更快。

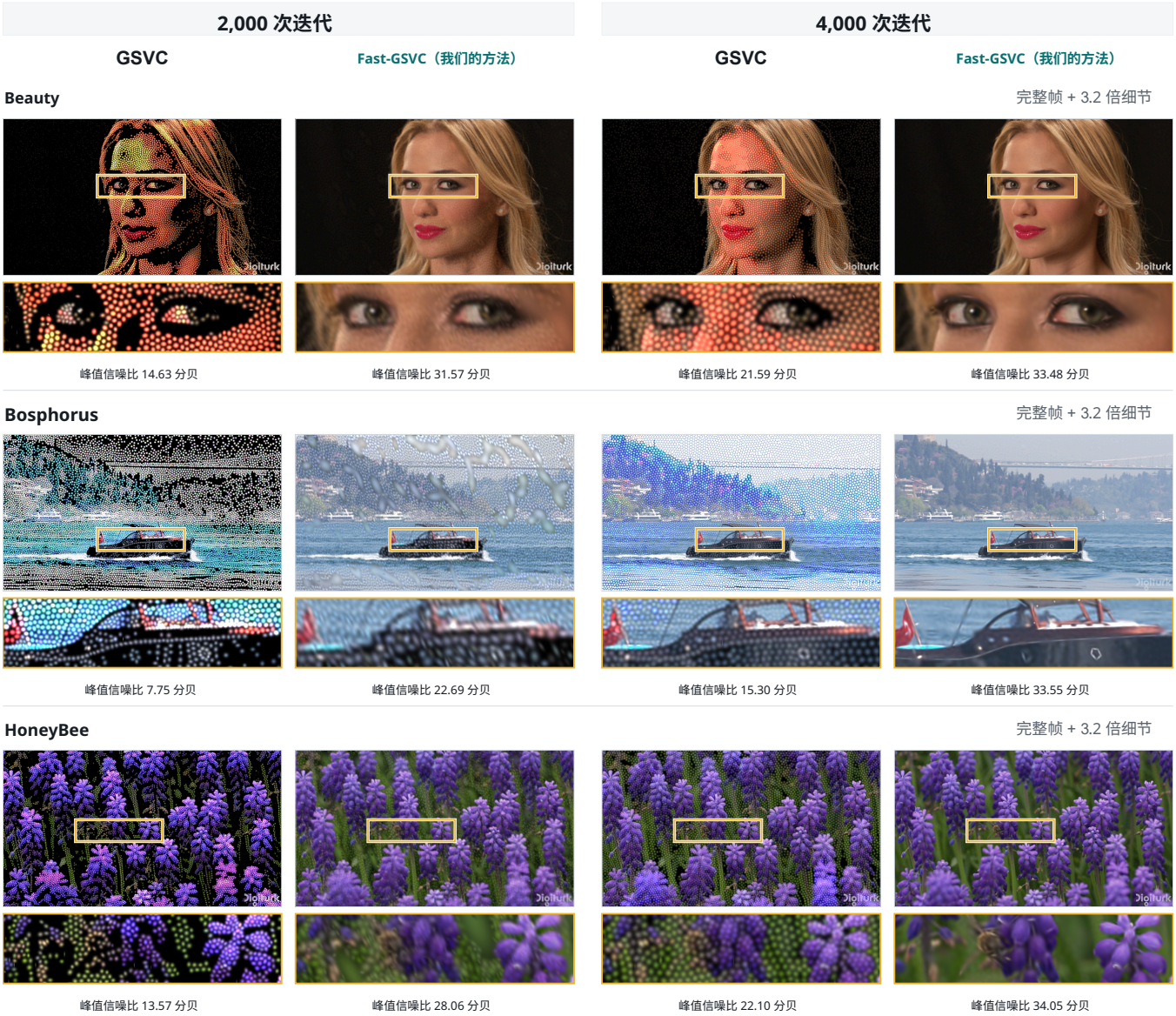


图 17：Beauty、Bosphorus 和 HoneyBee 在 2,000 次（左）和 4,000 次（右）迭代后的收敛对比。每组对比 GSVC 和 Fast-GSVC，下方展示峰值信噪比（分贝）和放大细节。所有重建结果均使用 10,000 个高斯。

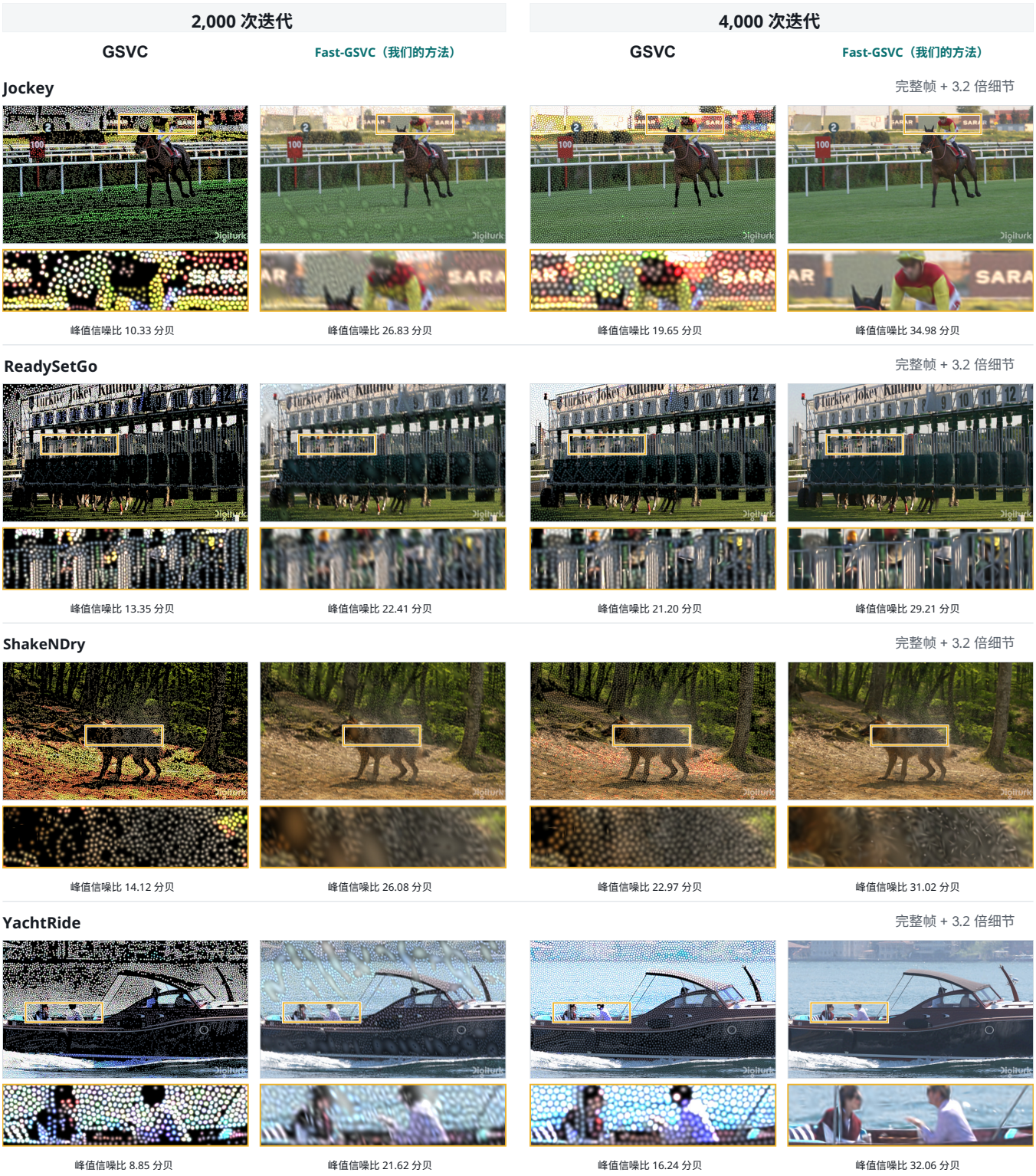


图 18：Jockey、ReadySetGo、ShakeNDry 和 YachtRide 在 2,000 次（左）和 4,000 次（右）迭代时的收敛性比较。每组比较 GSVC 和 Fast-GSVC，并在下方展示峰值信噪比（分贝）和放大细节。除由于动态渐进式调度，Fast-GSVC 在 2,000 次迭代的 ShakeNDry 使用 6,767 个高斯外，所有重建结果均使用 10,000 个高斯。

A.2 I帧质量

图19和图20比较了七个UVG序列中的I帧质量。尽管编码速度显著更快，Fast-GSVC仍能以相近或更低的比特率实现可比的重建质量，如重建帧和误差图所示。

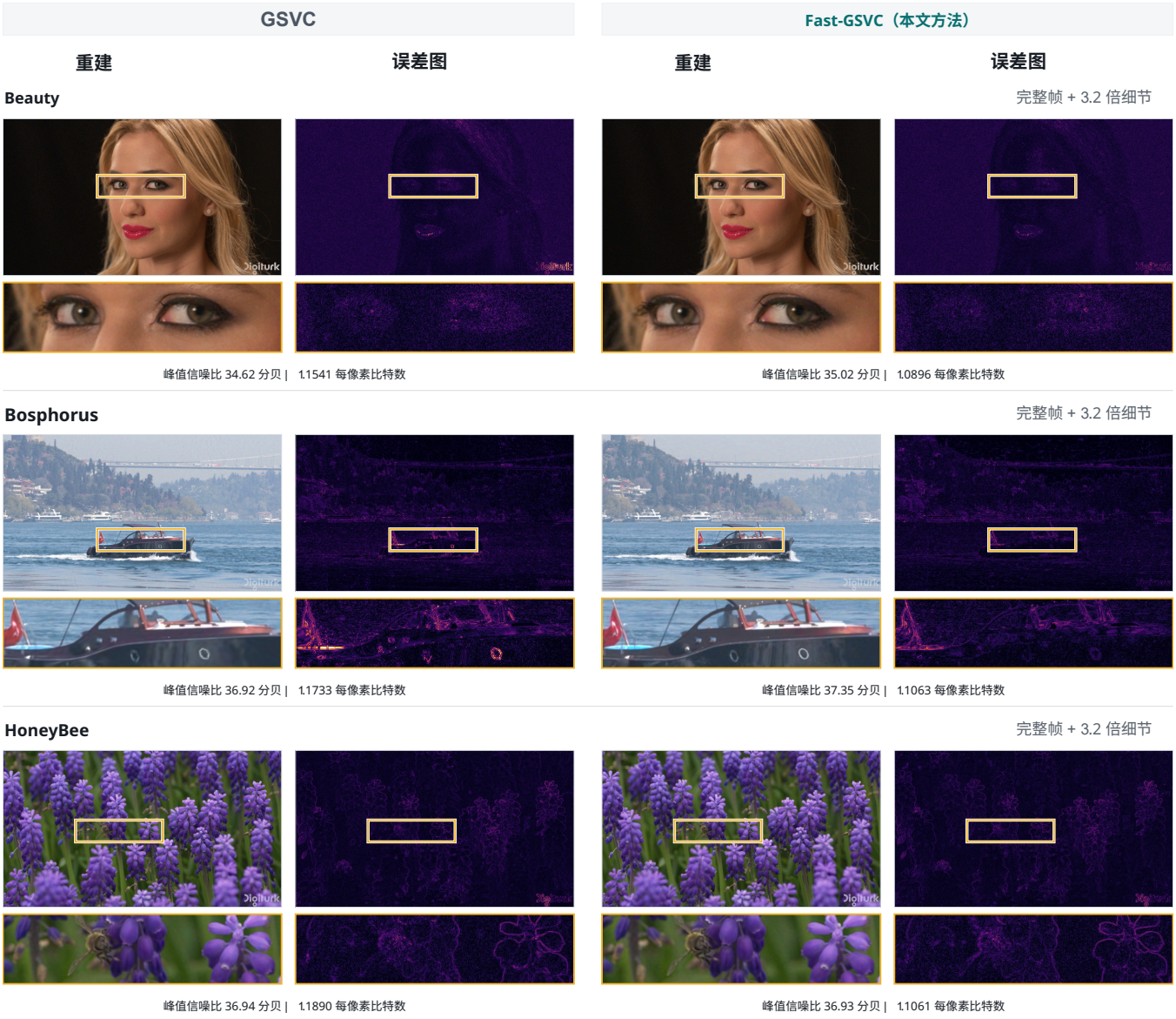


图19：Beauty、Bosphorus和HoneyBee上的I帧（第1帧）重建质量。通过重建结果、误差图和放大细节对GSVC与Fast-GSVC进行比较，并报告了每种方法的峰值信噪比（分贝）和每像素比特数。

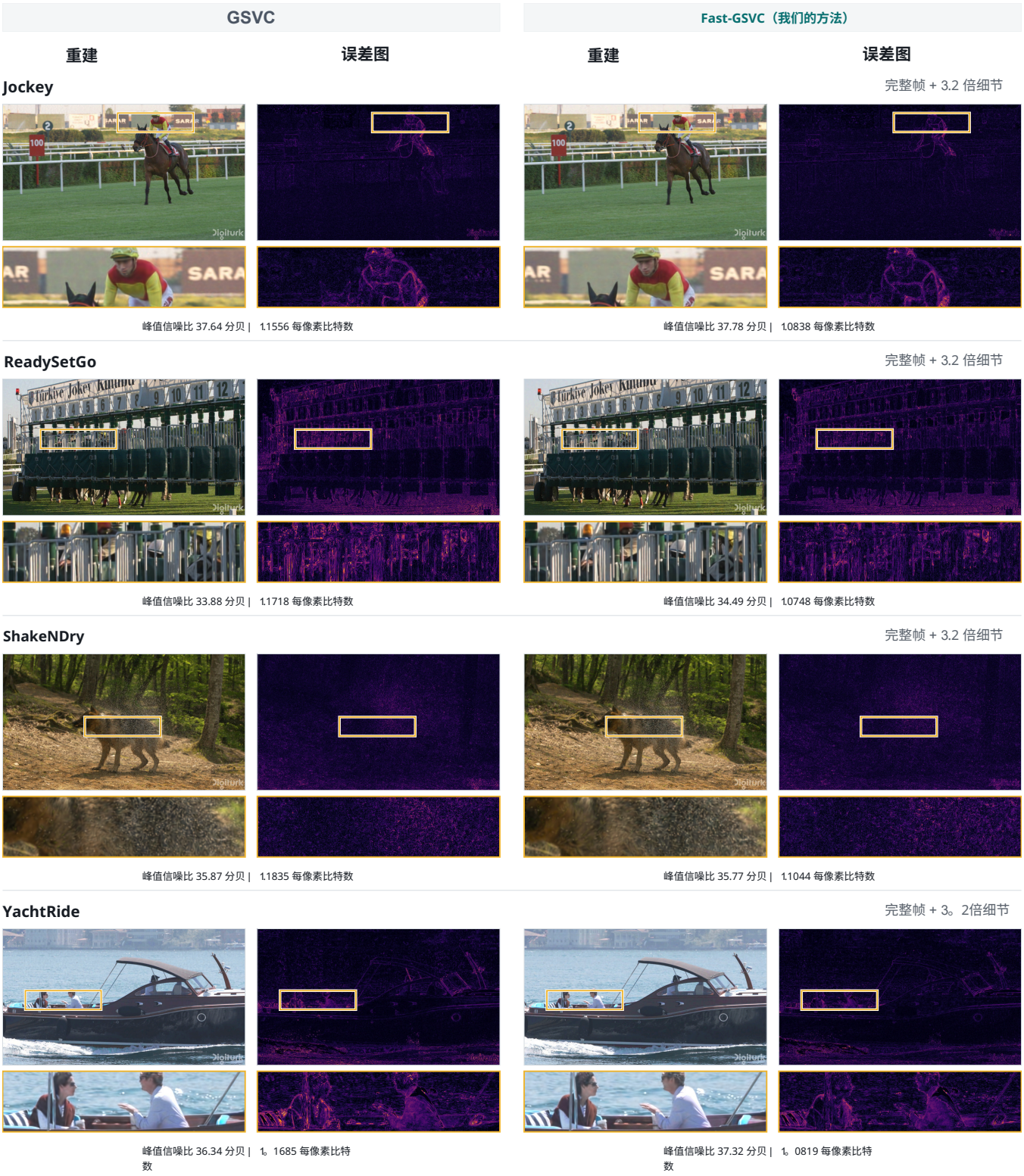


图20：Jockey、ReadySetGo、ShakeNDry 和 YachtRide 上 I帧（第1帧）的重建质量。通过重建结果、误差图和放大细节对 GSVC 与 Fast-GSVC 进行比较，并报告每种方法的峰值信噪比（分贝）和每像素比特数。

A.3 P帧质量

图21和图22比较了七个UVG序列中的P帧质量。尽管编码速度显著更快，Fast-GSVC在相近或更低的比特率下仍能实现更好或相当的重建质量，如重建帧和误差图所示。

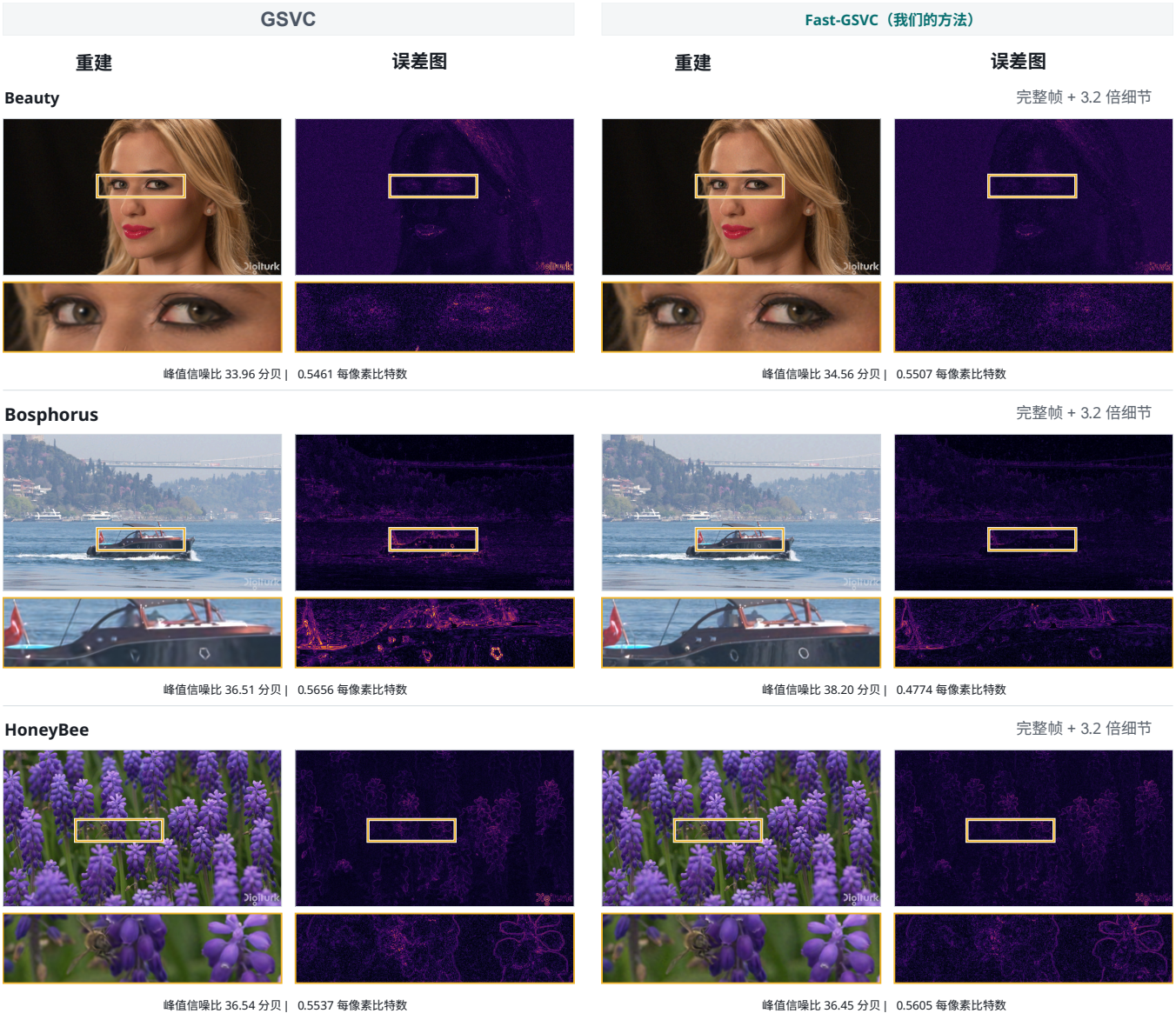


图21：Beauty、Bosphorus和HoneyBee上的P帧（第2帧）重建质量。通过重建结果、误差图和放大细节对GSVC与Fast-GSVC进行比较，并报告了每种方法的峰值信噪比（分贝）和每像素比特数。

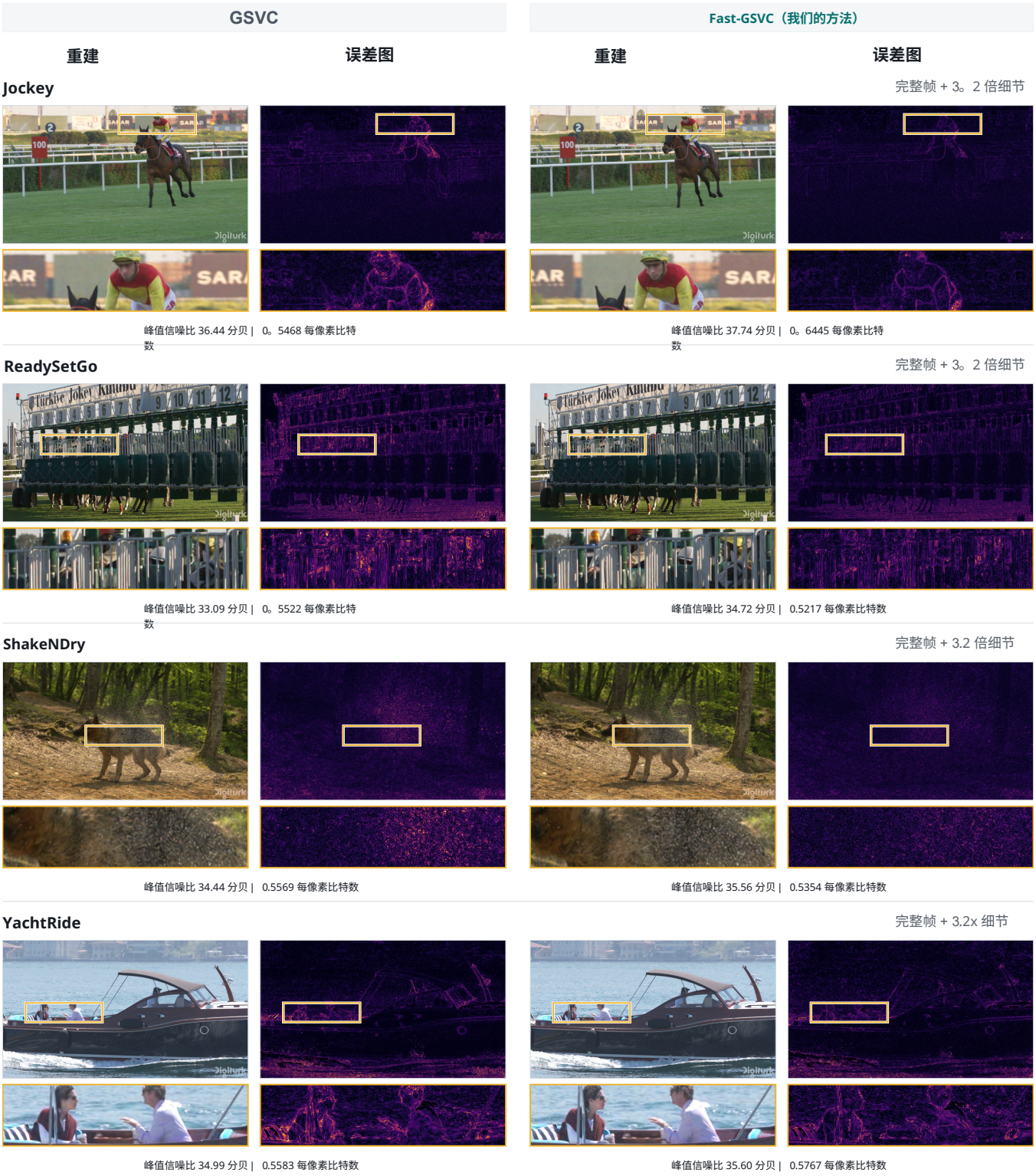


图22：Jockey、ReadySetGo、ShakeNDry 和 YachtRide 上 P帧（第2帧）的重建质量。通过重建结果、误差图和放大细节对比 GSVC 与 Fast-GSVC，并报告各方法的峰值信噪比（分贝）和每像素比特数。