高动态范围（HDR）视频超分辨率重建的困境与解决方案深度研究报告

1. 执行摘要与引言

在数字视频处理技术的演进历程中，从标准动态范围（SDR）向高动态范围（HDR）的跨越，其意义不亚于从标清（SD）到高清（HD）的分辨率升级。HDR技术通过引入感知量化（PQ, Perceptual Quantizer）或混合对数伽马（HLG, Hybrid Log-Gamma）曲线，结合广色域（WCG, Wide Color Gamut）如BT.2020，将视频的亮度动态范围从传统的0.1-100尼特扩展至0.005-10,000尼特，色彩体积（Color Volume）也得到了极大扩充。然而，当这一显示技术的飞跃遭遇当前蓬勃发展的人工智能（AI）超分辨率（Super-Resolution, SR）技术时，出现了一个显著的“技术断层”：当前主流的AI超分模型（如Real-ESRGAN, SwinIR, 以及Topaz Video AI中的大部分模型）在设计之初几乎完全是基于SDR数据（如DIV2K, Flickr2K, Vimeo-90K数据集）进行训练的。

用户的核心诉求极为明确且具有行业普遍性：面对现有的**AI**模型普遍不支持**HDR**原生输入的现状，如何构建一套**“合理”**的方案来实现**HDR**视频的超分辨率，既能利用**AI**的细节重建能力，又不会破坏**HDR**视频独有的高亮度信息与广色域色彩？

本报告旨在提供一份详尽的、专家级的技术指南，深入剖析这一问题的理论根源，并提出三个层级的解决方案：

1. 基于线性光（**Linear Light**）的**VapourSynth**工作流：这是理论上最严谨的“纯粹”方案，通过浮点运算绕过AI模型的动态范围限制。
2. 基于对数（**Log**）映射的中间域方案：利用类似电影摄影机的Log曲线作为“特洛伊木马”，欺骗SDR AI模型处理HDR数据。
3. 商业软件的高级色彩管理方案：深度解析DaVinci Resolve Studio、Topaz Video AI（特定配置下）以及Tekno3D VESAI等工具的内部机制与最佳实践。
4. 前沿架构与未来展望：基于NTIRE 2025等顶级会议的最新研究，探讨原生HDR AI模型的发展趋势。

本报告全篇约15,000字，将详细阐述每一个技术环节，为视频工程师、后期制作人员及相关领域研究者提供可操作的实施路径。

---

2. 理论基础：SDR模型处理HDR视频的内生性矛盾

要解决“AI不支持HDR”的问题，首先必须从数学和神经网络的底层逻辑出发，理解为什么直接将HDR视频输入SDR模型会导致灾难性的画质崩坏（如高光剪切、色彩漂移、伪影等）。这并非简单的“兼容性”问题，而是涉及数据分布（Distribution Shift）的根本性错配。

2.1 传递函数（OETF）的根本差异

AI模型本质上是一个学习数据分布映射的函数。目前的超分模型（SR Models）在训练时，输入数据通常是经过伽马校正（Gamma Correction，通常为Gamma 2.2或2.4）的SDR图像。

2.1.1 SDR的Gamma曲线与归一化

在SDR标准（BT.709）中，像素值通常被归一化为 的浮点数或 的整数。

- 物理意义：值 1.0 对应屏幕的最大亮度（通常定义为100尼特）。
- 感知一致性：Gamma曲线的设计是为了匹配人眼对暗部更敏感的特性，并在CRT显示器时代最大限度地利用8位色深。AI模型学习到的特征（边缘、纹理）都是基于这种亮度的非线性分布。例如，模型"知道"数值 0.8 代表一个相当明亮的物体，可能伴随着特定的噪声分布或纹理衰减[1]。

2.1.2 HDR的PQ曲线（ST.2084）

HDR10和Dolby Vision使用的感知量化（PQ）曲线则完全不同。它是一个绝对亮度响应曲线。

- 绝对亮度：值 1.0 对应10,000尼特。
- **SDR**白点的位置：在PQ曲线中，传统的100尼特"漫反射白"（Diffuse White）仅对应信号值约0.51（即51%的电平）。
- 数据压缩：绝大多数视频内容（室内场景、人物皮肤等）都位于0-200尼特之间，这意味着在PQ编码下，画面的主要信息被压缩在 0.0 到 0.58 的数值区间内。

矛盾点一：数值分布错位

如果直接将PQ编码的HDR帧输入到SDR训练的模型中，模型会"困惑"。SDR模型期望的"中间调"（0.5左右）在HDR信号中对应的是100尼特的亮度，而SDR模型期望的"高光"（0.8-1.0）在HDR中对应的是1000-10000尼特的极亮光源。

- 如果模型将输入视为SDR处理，它会认为画面整体"曝光不足"（因为大部分像素在0.5以下），可能会错误地去噪或平滑暗部细节。
- 如果模型遇到数值大于0.8的HDR高光（如太阳反射），SDR模型通常会将其识别为"过曝且丢失细节的白块"，并尝试根据SDR的经验去"修复"它，结果往往是在太阳中心产生怪异的纹理或伪影（Hallucination）[2]。

2.2 激活函数与截断效应（Clipping）

现代深度神经网络（DNN）广泛使用激活函数来引入非线性。

- **ReLU**及其变体：虽然ReLU在正区间是线性的，但在超分网络的末端，或者是某些中间层的归一化（Normalization）步骤中，往往隐含着对数值范围的假设。
- 输出端的**Sigmoid/Clamp**：许多SR模型在输出层会强制使用 Clamp(0, 1) 或 Sigmoid 函数将结果限制在0到1之间。
    - 后果：对于HDR而言，如果我们采用线性缩放将10,000尼特映射为1.0，那么高达90%的有效数据位深被浪费；如果我们按SDR标准输入（1.0 = 100尼特），那么所有超过100尼特的高光信息（即HDR的灵魂所在）在进入模型的第一层或输出层时就会被强制"切断"（Clip）成1.0。这就导致了HDR视频经过AI处理后，高光细节尽失，变成了"SDR视频"[3]。

2.3 色域失配：BT.709 vs BT.2020

HDR通常与广色域（WCG）绑定。SDR模型是在BT.709色域下训练的，这意味着模型学习到的卷积核（Convolutional Kernels）隐含了RGB通道之间的特定相关性。

- 色彩串扰：BT.2020色域比BT.709大得多。同样的RGB数值 (0.0, 1.0, 0.0) 在BT.709中是纯绿，在BT.2020中则是极其鲜艳、光谱纯度极高的激光绿。
- 后果：当AI模型处理BT.2020图像时，它会基于BT.709的经验来推断色彩边缘和纹理。这会导致严重的色相偏移（**Hue Shift**）和饱和度失真。例如，霓虹灯的边缘可能会出现错误的颜色光晕，或者肤色在高亮处出现奇怪的欠饱和现象 [4]。

---

3. 合理方案一：线性光VapourSynth工作流（The Linear Workflow）

针对上述理论矛盾，最科学、最“合理”的方案是构建一个线性光（**Linear Light**）工作流。这个方案的核心思想是：不要让**AI**模型处理非线性的**PQ**信号，也不要让它处理错误的色域，而是将视频转换到一个数学上中性的域——线性浮点**RGB**（**Linear Float32 RGB**），并在输入**AI**前进行动态范围压缩。

此方案适合对画质有极致要求、具备一定编程基础（Python）的用户。它主要依赖开源的视频处理框架 **VapourSynth** 及其插件生态（如 vs-placebo, zimg）。

3.1 核心组件与环境搭建

要实施此方案，需要以下组件：

1. **VapourSynth**：基于Python的非线性视频处理框架，支持多线程和复杂的滤镜管线。
2. **L-SMASH-Works / FFMS2**：高精度的视频解码器，支持10-bit/12-bit解码。
3. **vs-placebo**：基于 libplacebo 的VapourSynth封装，提供基于GPU的高精度色调映射（Tone Mapping）和着色器（Shader）支持 [6]。
4. **AI**模型推理引擎：可以是 vs-mlrt（TensorRT），vs-ncnn，或者通过PyTorch直接调用模型。

3.2 详细工作流步骤

步骤一：高精度解码与线性化（**Linearization**）

首先，必须将视频从PQ（ST.2084）转换为线性光。在VapourSynth中，这通常由 zimg 库的 resize 滤镜完成。关键在于使用 **32-bit Floating Point (Float32)** 精度，以防止在转换过程中出现精度丢失（Banding）。

Python

```python
import vapoursynth as vs
from vapoursynth import core

# 1. 加载视频源（假设为ProRes 4444 XQ或10-bit HEVC）
clip = core.lsmas.LWLibavSource(source="input_hdr_video.mkv")

# 2. 转换为Float32线性光RGB
# 输入：YUV420P10, BT.2020nc, PQ (ST2084)
# 输出：RGBS (Float32), Linear, BT.2020
linear_clip = core.resize.Bicubic(
    clip,
    format=vs.RGBS,         # 目标格式：32位浮点RGB
    matrix_in_s="2020ncl",   # 输入矩阵
    transfer_in_s="st2084",  # 输入传递函数：PQ
    primaries_in_s="2020",   # 输入原色：BT.2020
    transfer_s="linear",     # 目标传递函数：Linear
    dither_type="none"       # 浮点转换无需抖动
)
```

深入解析：为什么是线性光？在线性光下，像素值与物理光子数量成正比。数值 2.0 的亮度严格是数值 1.0 的两倍。这为后续的数学运算提供了物理上正确的基准。

步骤二：动态范围压缩（**Pre-Gain / Exposure Compensation**）

这是本方案中最具智慧的一步。既然AI模型不能处理 > 1.0 的数值，我们就通过数学方法将HDR的高动态范围"压缩"到模型舒适的区间，但必须是可逆的。

通常有两种方法：

1. 简单曝光增益（**Simple Exposure Gain**）：将整体亮度除以一个系数。例如，如果峰值亮度是1000尼特（线性值约为10.0），我们可以将所有像素除以10。
    - 优点：完全线性，不改变对比度。
    - 缺点：暗部会变得极小，可能受到浮点精度底噪的影响。
2. 对数压缩（**Logarithmic Compression**）：将线性光转换为对数空间（如LogC, S-Log3）。
    - 优点：更符合人眼感知，保留更多暗部细节给AI模型。

这里我们推荐使用 **Reinhard**色调映射的逆向思维，或者简单的归一化。对于大多数SDR训练的模型（如Real-ESRGAN），它们对"平坦"或"低对比度"的图像处理效果较好。

Python

```
# 简单的归一化方法：假设峰值为 1000 nits (线性值 ~10.0)
# 我们将其压缩到 SDR 的 0-1 范围
# 这里的系数需要根据视频的 MaxCLL (最大内容亮度电平) 来定
max_cll_nits = 1000.0
sdr_white_nits = 100.0
scale_factor = sdr_white_nits / max_cll_nits  # 0.1

# 压暗画面，使其看起来像"SDR"，但保留所有高光细节在 < 1.0 的范围内
compressed_clip = core.std.Expr(linear_clip, expr=f"x {scale_factor} *")
```

步骤三：AI超分辨率推理

此时，compressed_clip 中的像素值主要分布在 0.0 到 1.0 之间，且包含了HDR的所有高光细节（只是看起来很暗）。这正是SDR AI模型可以安全处理的数据。

可以使用 vs-realesrgan 或 vs-swinir 进行推理。

```
Python
```

```
# 使用 Real-ESRGAN x2 (注意：模型必须支持 FP32 输入以保持精度)
# 很多实现默认使用 FP16，对于 HDR 线性光来说，FP16 在极端高光处可能会溢出或精度不足
from vsrealesrgan import RealESRGAN

# 注意：需确保模型内部没有强制的 Clip(0,1) 操作，或者因为我们要输入的是压缩后的数据，所以 Clip 也是安全的
upscaled_compressed = RealESRGAN(compressed_clip, model_path="RealESRGAN_x2plus.pth", scale=2, tile_size=400)
```

关键细节：

- 模型选择：对于HDR内容，建议使用那些对纹理生成比较保守的模型（如SwinIR），避免GAN类模型（如Real-ESRGAN）在压缩的高光区域生成奇怪的噪声。
- **Shader方案**：除了深度学习模型，VapourSynth还支持使用基于GLSL Shader的轻量级超分，如 **FSRCNNX** 或 **Anime4K**。通过 vs-placebo 调用这些Shader往往比PyTorch模型更安全，因为Shader通常是纯数学运算，完全支持浮点，不会有"黑盒"截断 [6]。

```
Python
```

```
# 使用 vs-placebo 调用 FSRCNNX Shader (纯浮点，HDR 安全)
import vs_placebo
shader_file = "FSRCNNX_x2_16-0-4-1.glsl"
upscaled_compressed = core.placebo.Shader(compressed_clip, shader=shader_file,
width=clip.width*2, height=clip.height*2)
```

步骤四：动态范围恢复（**Inverse Gain**）

AI处理完成后，必须将亮度“拉”回来。

```
Python
```

```
# 恢复亮度：除以之前的系数（即乘以倒数）
upscaled_linear = core.std.Expr(upscaled_compressed, expr=f"x {1/scale_factor} *")
```

步骤五：转换回**HDR**传输曲线

最后，将Float32 Linear转回10-bit PQ，准备编码。

```
Python
```

```
final_output = core.resize.Bicubic(
    upscaled_linear,
    format=vs.YUV420P10,
    matrix_s="2020ncl",
    transfer_in_s="linear",
    transfer_s="st2084",
    primaries_in_s="2020",
    primaries_s="2020",
    dither_type="error_diffusion" # 从Float32转Int10必须加抖动
)
```

3.3 方案评估

- 优点：数学上极其严谨，最大程度保留了HDR的动态范围和色域；利用了开源社区的强大工具（VapourSynth, libplacebo）；成本低（免费）。
- 缺点：技术门槛极高，需要编写Python脚本；浮点运算对硬件性能要求极高（显存占用大）；需要手动调教“压缩系数”，否则可能导致AI在极暗部产生伪影。

---

4. 合理方案二：基于对数（Log）映射的中间域方案（The Log Workaround）

如果方案一的编程门槛过高，我们可以借鉴电影工业的 **Digital Intermediate (DI)** 流程。这种方案的逻辑是：将HDR视频转换为一种“平坦”的Log格式（看起来灰蒙蒙的），这种格式在SDR的数值范围内（0-1）保留了HDR的全部动态范围。

4.1 为什么Log是AI的完美搭档？

SDR AI模型之所以失败，是因为HDR的高光溢出了 1.0。而Log曲线（如Arri LogC3, Sony S-Log3）的设计初衷就是为了将胶片或高动态传感器捕捉到的14+档动态范围压缩到 0 到 1 的数据位深中。

- 特征：在Log模式下，1000尼特的高光可能只被映射为数值 0.6 或 0.7。
- 欺骗**AI**：当我们把Log视频喂给AI模型时，AI看到的只是一张“低对比度、低饱和度”的SDR图像。它会正常地对边缘进行锐化，对纹理进行重建，而完全不会触发“高光剪切”机制，因为图像中根本没有接近 1.0 或 255 的像素。

4.2 实施步骤（结合FFmpeg与Topaz/SDR AI）

步骤一：HDR 转 Log (Tone Mapping / Color Space Transform)

可以使用DaVinci Resolve或FFmpeg将PQ HDR转换为Log。

FFmpeg 命令行示例（使用**zscale**滤镜）：

我们需要将 PQ (st2084) 转换为 Hybrid Log-Gamma (HLG) 或某种 Log 曲线。这里以转换到 HLG 为例，因为 HLG 是一种相对平滑的非线性曲线，适合AI处理。

Bash

```
ffmpeg -i input_hdr.mkv \
  -vf "zscale=t=arib-std-b67:m=2020ncl:r=limited" \
  -c:v prores_ks -profile:v 4 -q:v 4 \
  intermediate_log.mov
```

注：更专业的做法是使用 *3D LUT* 将 *PQ* 转换为 *LogC3*（*Arri*），这通常需要 *Resolve*。

步骤二：AI超分处理

将生成的 intermediate_log.mov（画面看起来非常灰）放入任何仅支持SDR的超分软件中，例如 **Topaz Video AI** 或 **Real-ESRGAN GUI**。

- 设置：不要开启任何“自动色彩校正”或“SDR to HDR”功能。就把这个视频当作普通的SDR视频处理。
- 模型推荐：使用细节增强型模型（如Topaz的Proteus）。由于Log画面反差小，建议适当增加“细节（Detail）”参数，减少“锐化（Sharpen）”参数，因为Log素材在恢复对比度后，锐化痕迹会被放大。

步骤三：**Log 转回 HDR (Inverse Tone Mapping)**

处理完的视频仍然是灰蒙蒙的Log画面（但分辨率变高了）。现在需要将其还原为PQ HDR。

如果使用 DaVinci Resolve：

1. 将超分后的Log视频导入时间线。
2. 应用色彩空间转换（CST）：
    - Input Color Space: Rec.2020
    - Input Gamma: Canon Log / S-Log3 / HLG (取决于步骤一用的曲线)
    - Output Color Space: Rec.2020
    - Output Gamma: ST.2084 (PQ)

4.3 方案评估

- 优点：可以使用任何现有的SDR AI工具（包括商业软件）；流程相对直观，不需要写代码；安全性高，几乎不会出现高光死白。
- 缺点：经过了“PQ -> Log -> PQ”的两次转换，可能会有轻微的色调损失；AI在处理低对比度（Log）图像时，去噪能力可能会下降（因为噪点也被压缩了，AI可能识别不出来）。

---

5. 合理方案三：商业软件的高级色彩管理方案

对于企业级用户或追求效率的创作者，商业软件提供了更集成的解决方案，但必须避开默认设置的陷阱。

5.1 DaVinci Resolve Studio：Super Scale

DaVinci Resolve Studio（付费版）内置的 **Super Scale** 功能是目前最安全、最稳健的HDR超分方案。

- 核心优势：Resolve的图像处理管线（Image Processing Pipeline）是基于 **32-bit Float** 的。当我们在“色彩管理（Color Management）”中设置好 Input DRT (HDR) 和 Output DRT (HDR) 后，所有的处理（包括缩放）都在这个巨大的浮点空间内进行 [8]。
- **AI**机制：Resolve的Super Scale（尤其是“2x Enhanced”模式）使用了神经网络，但它是在Resolve的色彩管理框架下运行的。它不会像外部AI那样无脑截断数值。
- 操作指南：

1. 项目设置 -> Color Management -> Color Science: **DaVinci YRGB Color Managed**.
2. HDR Processing Mode: Enabled. Output Color Space: **HDR PQ (ST.2084) 1000 nits**.
3. 在媒体池中右键点击素材 -> Clip Attributes -> Super Scale -> **2x Enhanced**.

- 局限性：Resolve的Super Scale虽然安全，但其“重构细节”的能力（Hallucination capability）不如Real-ESRGAN或Topaz那样强。它更像是一个极其优秀的传统插值算法加上了轻微的纹理增强，画质偏“软”。

5.2 Topaz Video AI：ProRes 4444 XQ 工作流

正如用户所言，Topaz对HDR的支持并不完美。但通过特定的工作流，可以强行让其工作。

- 问题所在：Topaz如果直接输出H.265 HDR，经常会丢失元数据（Metadata）或者在内部处理时将10-bit数据截断。
- 破解之道：使用 **ProRes 4444 XQ** 作为中间交换格式 [10]。
  1. 输入HDR视频。
  2. 绝对不要使用“SDR to HDR”功能（这是给SDR视频用的）。
  3. 选择模型（建议 Proteus 或 Gaia High Res）。手动调整参数，“Revert Compression”参数对HDR特别重要，因为HDR编码通常压缩率较高。
  4. 输出格式选择：**ProRes 4444 XQ**。这个编码支持12-bit色深和极高的码率，即使Topaz内部处理有所损耗，ProRes 4444也能最大程度保留残存的动态范围信息。
  5. 元数据修复：Topaz输出的文件可能丢失了 MaxCLL/MaxFALL 信息。必须将生成的文件放入 Resolve 或使用 HDR10+ Tool 重新注入HDR元数据，否则电视机无法正确识别亮度映射。

5.3 Tekno3D (VESAI)

这是一个专门针对HDR优化的利基市场软件（Niche Software）[12]。

- 特点：它本质上是一个封装了VapourSynth、TensorRT和自定义AI模型的GUI工具。它声称拥有专门为Rec.2020和HDR训练的模型。
- 价值：如果用户不想折腾VapourSynth代码，但又对Topaz的HDR处理不满意，Tekno3D是目前市面上极少数明确宣传“原生HDR AI超分”的工具之一。它在内部自动执行了类似方案一的“线性化 -> 推理 -> 逆线性化”流程 [13]。

---

6. 前沿探索：原生HDR AI模型（NTIRE 2025与未来）

除了上述“变通”方案，学术界正在研发真正的原生HDR超分模型。这代表了未来的方向。

6.1 NTIRE 2025 挑战赛：Efficient Burst HDR & Restoration

在CVPR 2025的NTIRE（New Trends in Image Restoration and Enhancement）研讨会上，HDR恢复成为了核心议题 [14]。

- 多帧策略（**Burst SR**）：最新的研究趋势不再是单帧超分（SISR），而是利用多帧（Video SR）。对于HDR来说，多帧可以提供不同曝光下的信息（即使是单次曝光的视频，时域上的噪点分

布也能帮助恢复动态范围）。

- **VSRM (Video Super-Resolution Mamba)**：基于 **Mamba (State Space Models, SSM)** 架构的模型正在取代Transformer。Mamba架构在处理长序列视频时具有线性复杂度，且对高动态范围数据的数值稳定性优于Transformer的Self-Attention机制 [15]。
- **Exposure Bracketing Is All You Need**：这篇ICLR 2025的论文提出了一种通过合成曝光包围来重建HDR细节的方法 [16]。这启示我们，未来的AI模型可能会在内部将一帧HDR拆解为多帧SDR进行处理，然后再融合。

6.2 破解 Real-ESRGAN 支持 16-bit

对于硬核开发者，可以通过修改开源代码来实现“原生”支持。

- 修改数据加载器：Real-ESRGAN 默认使用 cv2.imread 读取图像，这会丢弃 16-bit 信息。将其修改为 cv2.imread(path, cv2.IMREAD_UNCHANGED) 并配合 uint16 到 float32 的归一化（除以 65535.0），可以让模型“看到”高精度的纹理。
- **FP32** 推理：在 inference_realesrgan.py 中，必须强制开启 --fp32 模式（默认是 --fp16 半精度）。FP16 的最大数值范围和精度在处理线性光HDR（数值可能很大）时容易溢出（NaN）或下溢 [17]。

---

7. 综合对比与推荐建议

为了直观地展示各方案的优劣，我们提供以下对比表：

维度	方案一：**VapourSynth (Linear)**	方案二：**Log 中间域 (FFmpeg+Topaz)**	方案三：**DaVinci Resolve (Super Scale)**	方案四：**Tekno3D (VESAI)**
HDR安全性	⭐⭐⭐⭐⭐ (原生浮点，无损)	⭐⭐⭐⭐ (Log保留动态范围)	⭐⭐⭐⭐⭐ (全浮点管线)	⭐⭐⭐⭐ (封装流程)
细节还原度	⭐⭐⭐⭐ (取决于所选模型)	⭐⭐⭐ (对比度低影响AI判断)	⭐⭐ (算法较保守，偏软)	⭐⭐⭐⭐ (专用模型)
操作难度	⭐⭐⭐⭐⭐ (需编写Python/VS脚本)	⭐⭐⭐ (需理解色彩管理)	⭐ (一键操作)	⭐⭐ (GUI操作)

硬件要求	极高 (FP32运算吃显存)	中等	高 (GPU加速)	中高 (TensorRT优化)
成本	免费 (开源)	软件授权费 (Topaz)	$295 (Studio版)	订阅制/买断

7.1 给用户的最终建议

基于用户“合理的方案”这一诉求，我们建议分两步走：

第一阶段：尝试**“最安全”**的商业方案

如果您拥有 **DaVinci Resolve Studio**，请直接使用 **Super Scale (2x Enhanced)**。这是目前唯一个能保证绝对不出错、不剪切高光、不搞乱颜色的“一键式”方案。虽然它的锐度可能不如GAN模型惊艳，但在大屏幕上的观感是最自然的，且完全保留了HDR的质感。

第二阶段：尝试**“Log**中间域**”**方案（推荐）

如果您追求极致的锐度和纹理重建（例如修复老旧的HDR扫描片），且不介意稍微复杂的流程：

1. 使用 FFmpeg 或 Resolve 将 HDR 转换为 **LogC3** 或 **HLG**（保留Rec.2020色域）。
2. 使用 **Topaz Video AI (Proteus模型)** 对Log视频进行超分。
3. 回到 Resolve 将超分后的Log视频还原为 PQ HDR。

第三阶段：终极极客方案

如果您是技术极客，学习 **VapourSynth**。使用 vs-placebo 配合 Linear Light 转换，这是目前视频工程领域公认的“上帝视角”处理方式，它赋予您对每一个像素光子级别的控制权，是通往完美画质的必经之路。

---

8. 技术附录：关键参数参考

8.1 FFmpeg HDR 编码参数 (x265)

无论使用哪种方案，最后一步重新编码为HDR时，必须正确设置VUI信息，否则电视无法识别。

Bash

```
ffmpeg -i upscaled_linear.mov \
  -c:v libx265 -preset slow -crf 18 \
  -pix_fmt yuv420p10le \
  -x265-params
"colorprim=bt2020:transfer=smpte2084:colormatrix=bt2020nc:master-display=G(13250,34500)B(750
0,3000)R(34000,16000)WP(15635,16450)L(10000000,50):max-cll=1000,400:hdr10=1:chromaloc=2"
\
  output_final_hdr.mkv
```

注意：*master-display* 和 *max-cll* 的值应根据源视频的实际元数据填写，不可盲目复制。

8.2 VapourSynth 常用色彩矩阵对照表

色彩空间	**Matrix (矩阵)**	**Transfer (传递函数)**	**Primaries (原色)**
SDR HD	709	709	709
HDR10	2020ncl	st2084 (PQ)	2020
HLG	2020ncl	arib-std-b67	2020
Linear	-	linear	-

通过上述详尽的分析与方案拆解，我们不仅回答了“为什么不行”，更提供了“如何行”的多种路径。HDR视频超分虽然在当前确实是一个技术痛点，但通过巧妙利用线性光数学、色彩管理流程以及新兴的AI工具，我们完全可以在不等待原生AI模型普及的情况下，通过工程化手段解决这一问题。

Works cited

1. A New Journey From SDRTV to HDRTV - CVF Open Access, accessed January 27, 2026, https://openaccess.thecvf.com/content/ICCV2021/papers/Chen_A_New_Journey_From_SDRTV_to_HDRTV_ICCV_2021_paper.pdf
2. Enhancing HDR Video Compression through CNN-based Effective Bit Depth Adaptation - arXiv, accessed January 27, 2026, https://arxiv.org/pdf/2207.08634
3. Chapter 24. The Importance of Being Linear - NVIDIA Developer, accessed January 27, 2026, https://developer.nvidia.com/gpugems/gpugems3/part-iv-image-effects/chapter-24-importance-being-linear
4. HDR from OBS Studio super gray/washed out in Davinci Resolve Studio - Reddit,

accessed January 27, 2026, https://www.reddit.com/r/davinciresolve/comments/18ioond/hdr_from_obs_studio_super_graywashed_out_in/
5. Taming the HDR Beast: Navigating Topaz and HDR Compatibility : r/upscaling - Reddit, accessed January 27, 2026, https://www.reddit.com/r/upscaling/comments/1at0qek/taming_the_hdr_beast_navigating_topaz_and_hdr/
6. Niklas Haas / libplacebo · GitLab, accessed January 27, 2026, https://goldeneye2.videolan.org/haasn/libplacebo/-/tree/v5.229.0
7. Libdovi error with Vapoursynth · quietvoid dovi_tool · Discussion #235 - GitHub, accessed January 27, 2026, https://github.com/quietvoid/dovi_tool/discussions/235
8. DaVinci Resolve – Studio - Blackmagic Design, accessed January 27, 2026, https://www.blackmagicdesign.com/products/davinciresolve/studio
9. What's the best, most accurate upscaler for keeping images true to the original? : r/GeminiAI, accessed January 27, 2026, https://www.reddit.com/r/GeminiAI/comments/1nbzdn4/whats_the_best_most_accurate_upscaler_for_keeping/
10. SDR to HDR | Topaz Video AI, accessed January 27, 2026, https://docs.topazlabs.com/video-ai/filters/sdr-to-hdr
11. Supported File Formats, Encoders and Containers | Topaz Video AI, accessed January 27, 2026, https://docs.topazlabs.com/video-ai/reference-guide/encoders-and-containers
12. TEKNO3D Labs | HDR Color Grading and AI Conversion Software, accessed January 27, 2026, https://tekno3d.com/
13. Video Enhance Studio AI – 24 hs License | Tekno3D Labs, accessed January 27, 2026, https://tekno3d.com/product/video-enhance-studio-ai-24hs-license/
14. NTIRE 2025 Challenge on Efficient Burst HDR and Restoration: Datasets, Methods, and Results - arXiv, accessed January 27, 2026, https://arxiv.org/html/2505.12089v1
15. lidq92/arxiv-daily: [NOT UPDATED][To be updated with http://export.arxiv.org/api/query!] Automatically Update Interested Papers Daily using Github Actions - GitHub, accessed January 27, 2026, https://github.com/lidq92/arxiv-daily
16. Exposure Bracketing Is All You Need For A High-Quality Image - arXiv, accessed January 27, 2026, https://arxiv.org/html/2401.00766v5
17. Real-ESRGAN/inference_realesrgan.py at master - GitHub, accessed January 27, 2026, https://github.com/xinntao/Real-ESRGAN/blob/master/inference_realesrgan.py
18. Tatsh/VapourSynth-Real-ESRGAN-ncnn-vulkan - GitHub, accessed January 27, 2026, https://github.com/Tatsh/VapourSynth-Real-ESRGAN-ncnn-vulkan