ChatGPT

LDR→HDR Conversion, Luminance-Stacking and Temporal Consistency

1 Background and Motivation

In VFX and computer-graphics pipelines it is common to work in **scene-referred linear space** and to store high-dynamic-range data in floating-point EXR files. A 16-bit half-float EXR can encode ~32 stops of dynamic range [1], which far exceeds the 8-stop range of a single digital exposure. Converting 8-bit LDR images into 32-bit HDR is desirable because it gives artists more headroom to grade shots, perform physically based lighting and avoid clipping. The provided `train_8bit2hdr.py` script trains a residual encoder–decoder (HDRNet-style) on aligned LDR–HDR pairs to map linearized 8-bit inputs to linear HDR outputs. It uses a hybrid loss that mixes linear and logarithmic terms, random tiling for training and optional mixed precision.

Some community pipelines (e.g., ComfyUI's *Img2HDRI* workflow) use a **Luminance Stack Processor** to build an HDR image from a single LDR panorama. This node splits the luminance of one image into several exposure levels, processes them individually and merges them to produce a synthetic exposure stack. The user asked whether such luminance-stacking could be used in place of deep learning and how to reduce frame-to-frame noise in sequences.

2 Existing HDR Reconstruction Approaches

2.1 Stack-based HDR reconstruction

Traditional HDR photography captures a *stack* of images at different exposure times and merges them into one irradiance map. After radiometric calibration, pixel values from the stack are linearly combined to estimate the true scene irradiance [2]:

$$\hat{E}(p) = \frac{\sum_i w_i \, X_i(p)/t_i}{\sum_i w_i} \tag{1}$$

where $X_i(p)$ is the linearized pixel value, t_i the exposure time and w_i a weight. Debevec & Malik proposed a triangular weight that attenuates saturated or under-exposed pixels [3]; Mann & Picard and Mitsunaga & Nayar derive weights based on the derivative of the camera response and the signal-to-noise ratio [4]. Proper weighting is important because images in the stack have different amounts of quantization noise, photon shot noise and thermal noise [5]. These methods can produce high-quality HDR for static scenes, but they require multiple exposures and suffer from ghosting when subjects move.

2.2 Radiometric calibration and single-image methods

Before merging, LDR images must be linearized by inverting the camera response function (CRF). Radiometric calibration is usually performed from multiple exposures, but **single-image methods** also exist. Matsushita & Lin exploit the fact that camera noise distributions should be symmetric; deviations from

symmetry indicate a nonlinear CRF, and the inverse CRF is recovered by restoring symmetry [6]. Lin et al. use edge linearity to estimate the CRF from a single image [7]. These techniques are accurate enough for moderate dynamic range but struggle with severe clipping.

2.3 Single-image HDR reconstruction via multi-exposure generation

Recent deep-learning work reconstructs HDR from **a single LDR image** by generating a synthetic exposure stack and then fusing it. Le et al. (2023) propose a weakly supervised network that inverts the camera response to recover pixel irradiance and then synthesizes multiple exposures; it hallucinates missing details in over- and under-exposed regions and merges them into HDR [8]. Chen et al. (2023) generalize this idea by learning a *continuous exposure value representation (CEVR)*: an implicit neural function generates LDR images with arbitrary exposure values (EVs) without ground-truth HDR [9]. Their model can produce a dense exposure stack (e.g., exposures at −2, −1.3, −0.7, 0, 0.7, 1.3 EV) leading to better HDR reconstruction [10]. These methods avoid ghosting and can work when only one image is available. However, they rely on learned hallucination and may not recover physically accurate radiance in heavily saturated regions.

2.4 Luminance-Stack Processor

Community tools such as ComfyUI's *Luminance Stack Processor* operate on a single LDR image. The node splits the image's luminance into several ranges (bright, mid-tone, dark), stretches each range (simulating different exposures), and recombines them into a synthetic HDR stack that can be exported as a 32-bit EXR. Conceptually this is similar to the multi-exposure-generation networks above but implemented with image processing rather than learned hallucination. Because it works on one image, the technique is sometimes mis-spelled "Luminance Stack Ptocessor" in online forums. It is useful when no HDR training data are available, but the dynamic range of the result is limited by the information in the input LDR.

3 Is luminance-stacking suitable for the current approach?

3.1 Potential benefits

1. **Works with a single LDR input.** If you only have AI-generated JPEGs with no matching HDR, synthetic exposure generation can create training pairs. Networks such as the CEVR model generate dense exposure stacks from a single LDR and then use Debevec's merging algorithm [9], which might simplify data acquisition.
2. **Handles dynamic scenes or panorama content.** Multi-exposure capture cannot be used for moving subjects; generating exposures from a single frame avoids ghosting.
3. **Data augmentation.** Even when you have LDR/HDR pairs, augmenting the LDR with synthetic exposures can help the network learn to expand dynamic range.

3.2 Limitations

1. **Limited dynamic range and detail.** A single 8-bit LDR contains clipped highlights and crushed shadows; synthetic exposures cannot recover information that is not present. Le et al.'s network hallucinates missing detail [8], which is acceptable for photographic images but may be unsuitable for physically accurate VFX lighting where linear radiance matters.

2. **Color shifts and non-physical mapping.** Splitting and stretching luminance can alter colour ratios and produce unrealistic colours, violating the scene-referred assumption required in ACES pipelines [11].
3. **Quality depends on weight selection.** Traditional HDR merging emphasises weighting functions to handle noise and clipping [12]; simple luminance splitting may ignore these noise models and lead to artifacts.

3.3 Recommendation

Luminance-stacking is a pragmatic option when no HDR data exist, but for **production-level VFX** the goal is to preserve physically plausible radiance. The training script you provided already learns a mapping from linearized 8-bit images to HDR using ground-truth EXRs. That approach retains more detail and can generalize across similar frames. If additional data are needed, consider combining both strategies: use a multi-exposure-generation network (CEVR) to augment training data and allow the network to learn exposure–irradiance relationships, but always validate outputs against physically correct HDR renders.

4 Minimising frame-to-frame noise and ensuring temporal consistency

4.1 Why temporal consistency matters

Image-processing models trained on static images can produce **temporal instability** when applied frame-by-frame to video. In low-light enhancement research, Zhang et al. observe that existing methods trained on single images "suffer serious temporal instability" when applied to video because no temporal information is available during training [13]. They propose learning motion priors (optical flow) from static images to synthesise short-range video sequences and impose consistency [14]. Similarly, in the general setting of blind video temporal consistency, applying an image processing algorithm independently to each frame results in undesirable flickering [15]. Lei et al. show that a convolutional network trained on the processed video with **Deep Video Prior** can reduce flickering without any handcrafted regularisation [16].

4.2 Strategies to reduce frame-to-frame noise

Below are techniques that can be integrated into your pipeline to minimise temporal noise:

1. **Temporal consistency loss.** When training on sequential frames, compute optical flow between adjacent frames, warp the previous HDR prediction to the current frame and penalize differences. This encourages the network to produce similar HDR values for corresponding pixels. The optical-flow-based approach used in low-light enhancement learns motion priors from static images [14]; the same idea can be applied here.

2. **Sequence-aware architectures.** Use 3D convolutions or recurrent units (Conv-LSTM) that process multiple frames jointly. These architectures naturally learn temporal dependencies and smooth out noise.

3. **Deep Video Prior or iterative refinement.** After generating HDR frames independently, apply a separate temporal-consistency network as a post-process. Lei et al. train a network directly on the

processed video to minimise the difference between the output and the processed frames while enforcing patch-wise consistency across time [16]. This approach does not require large datasets and can be applied to each sequence individually.

4. **Consistent data augmentation.** When training on synthetic sequences, apply the same random crop or augmentation to a contiguous block of frames so that the network learns to produce similar outputs for slightly shifted inputs. In your dataset loader, you can group consecutive frames and use consistent cropping parameters across them. Combining this with optical-flow-based loss helps the network generalize to motion.

5. **Noise-aware weighting.** In the HDR literature, Granados et al. argue that weights used for merging exposures should account for both spatial and temporal noise [17]. Extending this idea to deep learning, you can design the loss to down-weight uncertain pixels (e.g., saturated regions) and emphasise stable regions across frames.

6. **Denoising or film-grain modelling.** If the input LDRs exhibit noise, incorporate denoising (e.g., a residual network trained to remove sensor noise) before HDR conversion. Alternatively, model film grain and use noise injection during training to make the network robust to noise variations. This can reduce frame-to-frame noise due to random pixel fluctuations.

4.3 Implementation suggestions for the provided script

The current training pipeline (`train_8bit2hdr.py`) processes each frame independently and randomly crops patches for training. To improve temporal stability:

- **Extend the dataset class** to load sequences (e.g., lists of adjacent frames) instead of single images. Provide a method that returns a batch of consecutive frames and corresponding HDR targets with identical random crop coordinates.

- **Add a temporal loss term** to `hdr_loss`. For two consecutive predictions $\hat{Y}_t$ and $\hat{Y}_{t-1}$ and optical flow $F_{t\to t-1}$, compute a consistency loss $\|\hat{Y}_t - \mathcal{W}(\hat{Y}_{t-1}, F_{t\to t-1})\|_1$ where $\mathcal{W}$ warps the previous prediction. Balance this term with the existing HDR loss.

- **Use sliding-window inference** (the `enrich_ldr_sliding` function) for sequences to reduce patch-boundary artifacts and average predictions over overlapping windows.

- **Consider training a post-processing network** similar to Lei et al.'s deep video prior: feed the independently processed HDR frames and train a small network to minimise flicker.

5 Optimising the training pipeline for professional VFX workflows

5.1 Colour management and linear workflow

Professional VFX work requires a **scene-referred, linear colour pipeline**. John Daro emphasises that digital sensors have a linear response and convert this to logarithmic form for post-manipulation [18]; 16-bit EXR files can accommodate ~32 stops of range [1]. He notes that grading should be scene-referred—

maintaining the relationship to the original lighting—rather than display-referred [11]. Thus, your network's outputs should remain in linear radiance. Tone mapping should only be used for visualisation.

If you intend to integrate into ACES, ensure your network learns to map sRGB-to-linear correctly and that you apply the appropriate ACES Input Device Transform (IDT) to your LDR inputs. This will make your predictions consistent with other ACES-compliant assets.

5.2 Radiometric calibration and exposure matching

Because your input images come from various sources (PNG files converted from physical camera sequences or AI renders), it is important to calibrate them to a consistent radiometric response. The radiometric calibration section of Gallo & Sen explains that single-image methods may be used when multiple exposures are not available [19]. Calibrating the CRF ensures that similar pixel values correspond to similar scene irradiance across frames, reducing variation in the network's input. Where possible, convert all LDR images to linear light using a known CRF (e.g., sRGB inverse gamma or custom camera response) before training.

5.3 Data diversity and augmentation

Training data should cover a wide range of scenes, lighting conditions and exposure ranges. Include sequences of frames so the network learns temporal relationships. If you cannot capture HDR ground truth for every frame, consider using synthetically generated HDR (using physically based rendering) to provide supervised targets. Additionally, random transformations (flip, rotation, brightness shifts) should be applied consistently across frames to improve invariance.

5.4 Loss functions and evaluation

The current hybrid loss (linear + log) balances absolute radiance accuracy and relative differences. For HDR evaluation, compute metrics such as PSNR/SSIM on tone-mapped outputs. Consider using *HDR-VDP-2* or *HDR-VDP-3* metrics for perceptual quality. For temporal consistency, compute per-pixel standard deviation across frames to quantify flicker.

5.5 Output format and bit depth

Continue writing EXR files in half-float (`EXR_HALF`) or full-float if extreme intensities are expected. The 32-bit EXR format maintains linear radiance values and metadata for color space. Provide consistent naming to match LDR and HDR pairs during training.

5.6 Computational optimisations

- **Mixed precision and gradient accumulation.** The script already supports AMP (`--amp`) and gradient accumulation to reduce VRAM usage. Continue using these features, especially for large patches.
- **Channel-last memory layout.** Setting `--channels-last` can improve tensor cores throughput on some GPUs.
- **Efficient dataloaders.** Use multiple workers and pinned memory to speed up data loading.
- **Checkpoint management.** Continue to checkpoint the best model and resume training as needed.

6 Conclusion

High-quality HDR reconstruction is essential in VFX. Traditional stack-based HDR merges multiple exposures using radiometric calibration and noise-aware weights [20] [21]. Deep learning can directly map LDR to HDR; the provided script implements such a model and already operates in linear space. *Luminance-stacking* (splitting an LDR image into several exposure ranges and recombining them) is a useful technique when no HDR data are available, but it cannot recover information lost to clipping and may produce non-physical colours. State-of-the-art networks generate multi-exposure stacks from a single image [8] [9]; these approaches could augment your dataset but should be validated for physical accuracy.

For sequence processing, temporal inconsistency must be addressed. Processing each frame independently leads to flicker, as noted in low-light enhancement [13] and blind video temporal consistency research [15]. Incorporating temporal losses, sequence-aware architectures or post-processing methods (e.g., Deep Video Prior) can dramatically reduce frame-to-frame noise. Finally, maintaining a scene-referred linear workflow, calibrating the camera response and using EXR output will ensure your HDR images integrate seamlessly into professional VFX pipelines.

---

[1] [11] [18] HDR - Flavors and Best Practices — John Daro
https://www.johndaro.com/blog/2019/8/14/hdr-flavors-and-best-practices

[2] [3] [4] [5] [6] [7] [12] [17] [19] [20] [21] Title
https://research.nvidia.com/sites/default/files/pubs/2016-04_Stack-Based-Algorithms-for/Gallo-Sen_StackBasedHDR_2016.pdf

[8] Single-Image HDR Reconstruction by Multi-Exposure Generation
https://openaccess.thecvf.com/content/WACV2023/papers/Le_Single-Image_HDR_Reconstruction_by_Multi-Exposure_Generation_WACV_2023_paper.pdf

[9] [10] Learning Continuous Exposure Value Representations for Single-Image HDR Reconstruction
https://openaccess.thecvf.com/content/ICCV2023/papers/Chen_Learning_Continuous_Exposure_Value_Representations_for_Single-Image_HDR_Reconstruction_ICCV_2023_paper.pdf

[13] [14] Learning Temporal Consistency for Low Light Video Enhancement From Single Images
https://openaccess.thecvf.com/content/CVPR2021/papers/Zhang_Learning_Temporal_Consistency_for_Low_Light_Video_Enhancement_From_Single_CVPR_2021_paper.pdf

[15] [16] 0c0a7566915f4f24853fc4192689aa7e-Paper.pdf
https://proceedings.nips.cc/paper_files/paper/2020/file/0c0a7566915f4f24853fc4192689aa7e-Paper.pdf