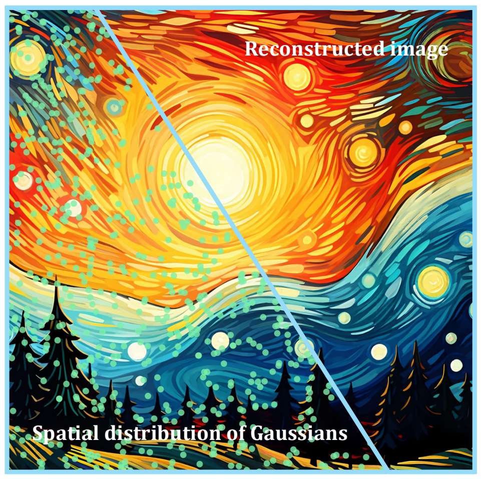
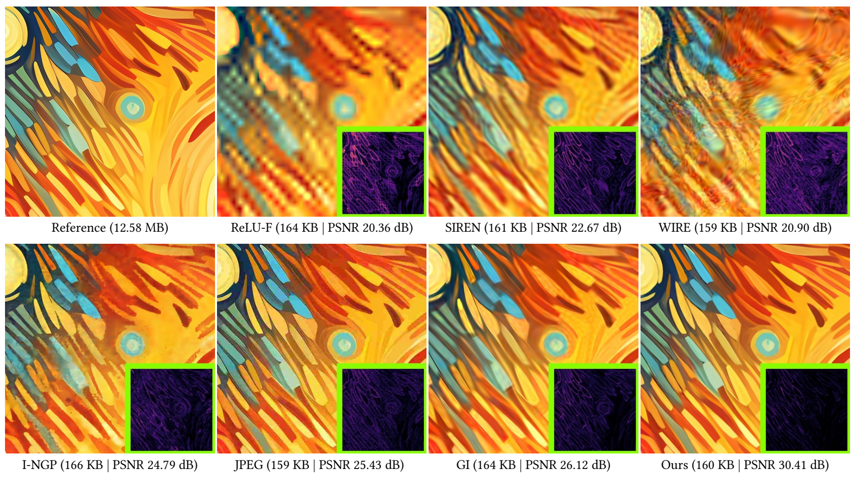
Image-GS: Content-Adaptive Image Representation via 2D Gaussians

YUNXIANG ZHANG* and BINGXUAN LI*, New York University, USA
ALEXANDR KUZNETSOV, Advanced Micro Devices, USA
AKSHAY JINDAL, Intel Corporation, USA
STAVROS DIOLATZIS, Intel Corporation, France
KENNETH CHEN, New York University, USA
ANTON SOCHENOV and ANTON KAPLANYAN, Intel Corporation, USA
QI SUN, New York University, USA




Reference (12.58 MB)
ReLU-F (164 KB | PSNR 20.36 dB)
SIREN (161 KB | PSNR 22.67 dB)
WIRE (159 KB | PSNR 20.90 dB)
I-NGP (166 KB | PSNR 24.79 dB)
JPEG (159 KB | PSNR 25.43 dB)
GI (164 KB | PSNR 26.12 dB)
Ours (160 KB | PSNR 30.41 dB)

(a) content-adaptive resource allocation (b) visual comparison against conventional and neural image representations

Fig. 1. *Image-GS* reconstructs an image by adaptively allocating and progressively optimizing a set of colored 2D Gaussians. It achieves favorable rate-distortion trade-offs, hardware-friendly random access, and flexible quality control through a smooth level-of-detail stack. (a) visualizes the optimized spatial distribution of Gaussians (20% randomly sampled for clarity). (b) Image-GS's explicit content-adaptive design effectively captures non-uniformly distributed image features and better preserves fine details under constrained memory budgets. In the inset error maps, brighter colors indicate larger errors.

Neural image representations have emerged as a promising approach for encoding and rendering visual data. Combined with learning-based workflows, they demonstrate impressive trade-offs between visual fidelity and memory footprint. Existing methods in this domain, however, often rely on fixed data structures that suboptimally allocate memory or compute-intensive implicit models, hindering their practicality for real-time graphics applications.

Inspired by recent advancements in radiance field rendering, we introduce Image-GS, a content-adaptive image representation based on 2D Gaussians. Leveraging a custom differentiable renderer, Image-GS reconstructs images by adaptively allocating and progressively optimizing a group of anisotropic, colored 2D Gaussians. It achieves a favorable balance between visual fidelity and memory efficiency across a variety of stylized images frequently seen in graphics workflows, especially for those showing non-uniformly distributed features and in low-bitrate regimes. Moreover, it supports hardware-friendly rapid random access for real-time usage, requiring only 0.3K MACs to decode a pixel. Through error-guided progressive optimization, Image-GS naturally constructs a smooth level-of-detail hierarchy. We demonstrate its versatility with several applications, including texture compression, semantics-aware compression, and joint image compression and restoration.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Rendering**; **Image compression**; **Image manipulation**.

Additional Key Words and Phrases: Image and texture representation

ACM Reference Format:
Yunxiang Zhang, Bingxuan Li, Alexandr Kuznetsov, Akshay Jindal, Stavros Diolatzis, Kenneth Chen, Anton Sochenov, Anton Kaplanyan, and Qi Sun. 2025. Image-GS: Content-Adaptive Image Representation via 2D Gaussians.

*Both authors contributed equally to this research.

Authors' Contact Information: Yunxiang Zhang, yunxiang.zhang@nyu.edu; Bingxuan Li, bingxuan.li@nyu.edu, New York University, Brooklyn, USA; Alexandr Kuznetsov, kuzsasha@gmail.com, Advanced Micro Devices, Bellevue, USA; Akshay Jindal, akshay.jindal@intel.com, Intel Corporation, Bellevue, USA; Stavros Diolatzis, stavros.diolatzis@intel.com, Intel Corporation, Nice, France; Kenneth Chen, kennychen@nyu.edu, New York University, Brooklyn, USA; Anton Sochenov, anton.sochenov@intel.com; Anton Kaplanyan, anton.kaplanyan@intel.com, Intel Corporation, Bellevue, USA; Qi Sun, qisun@nyu.edu, New York University, Brooklyn, USA.

ACM ISBN 979-8-4007-1540-2/2025/08
https://doi.org/10.1145/3721238.3730596

In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3721238.3730596

1 Introduction

Recent advances in generative AI have dramatically increased the availability of high-resolution visual content in domains ranging from gaming to graphic design [Epstein et al. 2023; Po et al. 2024]. Effectively deploying these assets, particularly to resource-constrained devices, requires representations that are compact and efficient to decode. Traditional image formats, such as PNG and JPEG, are often inadequate for this purpose, offering limited compression efficiency and/or slow decoding performance [Vaidyanathan et al. 2023].

Neural image representations are emerging as a promising alternative for encoding visual data [Chen et al. 2021; Tancik et al. 2020]. When integrated into learning-based pipelines, they enable superior visual fidelity and memory efficiency compared to classical formats. However, existing methods along this line often rely on fixed data structures that lack content adaptivity [Chen et al. 2023; Karnewar et al. 2022] or compute-intensive implicit neural models [Saragadam et al. 2023; Sitzmann et al. 2020], leading to poor scalability and slow decoding. These drawbacks hinder their usage in real-time graphics, where fast random access and dynamic quality adaptation to device capabilities are critical factors [Akenine-Moller et al. 2019].

To close this gap, we introduce Image-GS, an explicit image representation built on anisotropic, colored 2D Gaussians, each defined by a mean, a covariance, and a color. Given a target image, Image-GS adaptively initializes a group of Gaussians guided by local image gradient magnitudes, allocating more to higher-frequency regions. These parameters are then optimized using a custom differentiable renderer to reconstruct the image. Additional Gaussians are progressively added to regions with persistent artifacts to further refine the reconstruction quality. Image-GS's content-adaptive design captures the non-uniformly distributed features and semantic structures in images, allowing it to better preserve fine details under constrained memory budgets. To facilitate real-time usage, Image-GS's complete rendering pipeline is implemented with optimized CUDA kernels to maximize computational parallelism.

We evaluate the representation efficiency of Image-GS through extensive comparisons against recent neural image representations and industry-standard texture compressors across diverse images and textures. Image-GS demonstrates favorable trade-offs between visual fidelity and memory/computation cost, especially for graphics assets with non-uniformly distributed features and in low-bitrate scenarios. In addition, Image-GS supports fast parallel decoding and hardware-friendly random access, as well as flexible quality control via a smooth level-of-detail hierarchy. To showcase its versatility, we further employ Image-GS for two applications: semantics-aware compression and image restoration. Our source code and evaluation dataset are released at https://github.com/NYU-ICL/image-gs.

In summary, our main contributions include:

- a content-adaptive image representation supporting hardware-friendly random access and flexible rate-distortion trade-offs;
- a custom differentiable renderer optimized for efficient decoding;
- semantics-aware compression and image restoration applications.

2 Related Work

2.1 Traditional Image and Texture Compression

General-purpose image compression often prioritizes storage and transmission efficiency over decoding speed. Lossless approaches optimize pixel permutation and apply entropy coding [Welch 1985], while lossy ones encode image blocks using wavelet or cosine transforms [Antonini et al. 1992; Wallace 1991], followed by quantization. Advanced variants account for human visual sensitivity, higher bit depths, wide color gamuts, and user statistics [Alakuijala et al. 2019], along with content-adaptive block sizes and looped filtering [Chen et al. 2018]. Despite offering strong compression, these formats are slow to decode and poorly suited for multi-channel texture stacks.

In contrast, texture compression methods are designed to enable rapid decoding, support random access, and reduce GPU bandwidth. They operate on independent pixel blocks [Delp and Mitchell 1979], encoding per-pixel colors [Campbell et al. 1986], base colors with modifier values [Ström and Akenine-Möller 2005; Ström and Pettersson 2007], or color endpoints with interpolation indices [BC 2024]. Advanced schemes support HDR content, dynamic block sizes, and per-block adaptivity [Nystad et al. 2012], but their minimum bitrates are typically limited to around one bit per pixel (bpp).

2.2 Neural Image Representation and Compression

Neural image representations use neural models and deep features to efficiently encode visual media. Early methods employ autoencoders to transform images into compressed latent vectors and run training on large-scale image datasets to ensure generalizable performance [Ballé et al. 2017; Cheng et al. 2020; Theis et al. 2017]. More recent approaches overfit lightweight MLP models to individual images [Dupont et al. 2021, 2022], markedly reducing decoding complexity. Several works further enhance these MLPs by introducing explicit non-linearities, such as sinusoidal functions [Sitzmann et al. 2020], ReLU activations [Karnewar et al. 2022], positional encoding [Tancik et al. 2020], and Gabor wavelets [Saragadam et al. 2023], to better capture fine details. Advanced variants leverage per-image learned entropy models to improve compression [Ladune et al. 2023; Leguay et al. 2023]. Despite achieving strong rate-distortion performance, these models often demand lengthy training and compute-intensive decoding, limiting their practicality for real-time applications. For instance, C3 requires 3K MACs (multiply-accumulate operations) to decode a pixel at 0.31 bpp [Kim et al. 2024], while Image-GS requires only 0.3K MACs, an order of magnitude lower.

Another line of research explores hybrid models that combine neural decoders with explicit data structures carrying deep features to improve scalability [Martel et al. 2021], accelerate decoding [Müller et al. 2022], boost compression efficiency [Takikawa et al. 2022], and capture discontinuous signals [Belhe et al. 2023]. Vaidyanathan et al. [2023] extended the multi-resolution hash grid approach proposed in [Müller et al. 2022] for encoding multi-channel textures and their mipmap chains, achieving strong compression and rapid decoding. Yet, this method only supports a few fixed compression ratios and uses uniform feature grids optimized for material textures, limiting its efficiency on broader image types with non-uniformly distributed fine details. In contrast, Image-GS offers content adaptivity, flexible rate-distortion trade-offs, and fast random access simultaneously.

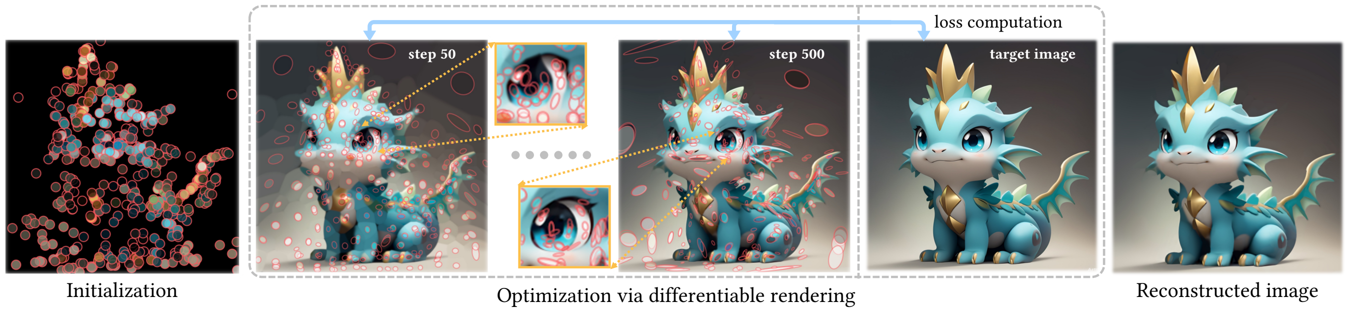


Fig. 2. *Image-GS optimization pipeline.* At initialization, a group of 2D Gaussians is adaptively spawned guided by local image gradient magnitudes, with more allocated to high-frequency areas (Section 3.3). During training, their parameters (Section 3.1) are optimized using a custom differentiable renderer (Section 3.2) to reconstruct the target, and additional Gaussians are progressively added to areas exhibiting persistent reconstruction errors (Section 3.3). 20% randomly sampled Gaussians are visualized as colored elliptical discs (scale and shape determined by the covariance) to illustrate the optimization progress.

2.3 Gaussian-based Representations

Parallel to neural representations, Gaussian-based representations are gaining traction in computer graphics. Our work is inspired by the recent success of Gaussian Splatting [Kerbl et al. 2023], where 3D Gaussians are employed for scene reconstruction and high-quality real-time rendering. Several follow-up works extended this method to support dynamic scenes [Luiten et al. 2024], on-the-fly training [Sun et al. 2024], surface modeling [Huang et al. 2024], and a variety of downstream applications [Fei et al. 2024]. Despite that Gaussian mixtures have been applied to image modeling [Celik and Tjahjadi 2011; Tu et al. 2024], restoration [Niknejad et al. 2015], compression [Sun et al. 2021; Zhu et al. 2022], and semantic segmentation [Ban et al. 2018], their potential as efficient image representation for real-time graphics remains largely underexplored. Concurrent with our work, GaussianImage [Zhang et al. 2024] also used 2D Gaussians to represent and compress images. While the basic building blocks are similar, our content-aware initialization and optimization strategies, coupled with top-K normalization during rendering, enable superior rate-distortion trade-offs. Moreover, GaussianImage relies on two-stage optimization and computationally intensive vector-quantization fine-tuning, making optimization and decoding speeds respectively 10× and 4× slower than ours at similar bitrates.

3 Method

3.1 Representing Images as 2D Gaussians

Similar to the Gaussian primitives in 3D Gaussian Splatting [Kerbl et al. 2023], a 2D Gaussian's geometry is defined by a mean vector $\boldsymbol{\mu} \in \mathbb{R}^2$ and a covariance matrix $\Sigma \in \mathbb{R}^{2\times2}$, and its value evaluated at an arbitrary pixel location $\mathbf{x} \in \mathbb{R}^2$ can be expressed as:

$$G(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{1}$$

To ensure that Σ remains positive semi-definite during numerical optimization, we instead work with its factorized form that consists of a rotation matrix $\mathbf{R} \in \mathbb{R}^{2\times2}$ and a scaling matrix $\mathbf{S} \in \mathbb{R}^{2\times2}$:

$$\Sigma = \mathbf{R}\,\mathbf{S}\,\mathbf{S}^T\,\mathbf{R}^T, \tag{2}$$

Specifically, we establish and maintain a rotation angle $\theta \in [0,\pi]$ and a scaling vector $\mathbf{s} \in \mathbb{R}^2_+$ for each 2D Gaussian. These attributes are updated via stochastic gradient descent and clipped to the valid ranges during training. The rotation and scaling matrices $\mathbf{R}, \mathbf{S}$ are constructed via $\theta, \mathbf{s}$ on the fly during both training and inference.

Unlike 3D Gaussian Splatting which models view-dependent appearance using spherical harmonics [Kerbl et al. 2023], we only need to associate each 2D Gaussian with a vector $\mathbf{c} \in \mathbb{R}^n$ to store its color, as an image essentially shows a single view of a 3D scene. Notably, the design choice of a variable color dimension n enables Image-GS to flexibly support a diversity of image formats, including grayscale, RGB, and CMYK images, as well as multi-channel texture stacks.

In addition, 3D Gaussian Splatting relies on an opacity attribute for depth-ordered occlusion and α-blended rendering. By contrast, the color information of 2D Gaussians can be effectively aggregated regardless of their relative order, as detailed in Section 3.2. Therefore, our 2D Gaussians do not have an opacity attribute.

By combining the above components, each 2D Gaussian primitive in Image-GS is fully characterized by $5+n$ trainable parameters:

$$\mathbf{p}_i \coloneqq \mathbf{p}_i(\boldsymbol{\mu}_i, \theta_i, \mathbf{s}_i, \mathbf{c}_i) \in \mathbb{R}^{5+n}, \quad 1 \le i \le N_g. \tag{3}$$

3.2 Rendering 2D Gaussians into Images

While 3D Gaussian Splatting requires opacity and depth sorting to handle occlusions and enforce multi-view consistency, we note that such requirements are unnecessary in the 2D case. In particular, an image showing a set of colored Gaussian blobs can be rendered by summing their weighted colors, and the resulting image is invariant to the order in which the Gaussians are applied.

Leveraging this insight, we simplify the point-based α-blending equation in [Kopanas et al. 2021; Yifan et al. 2019] by treating the 2D Gaussians as an unordered set of anisotropic points, and accumulate their color contributions to render an image pixel $\mathbf{c}_{\mathrm{r}}(\mathbf{x}) \in \mathbb{R}^n$:

$$\mathbf{c}_{\mathrm{r}}(\mathbf{x}) = \sum_{i=1}^{N_g} G_i(\mathbf{x}) \cdot \mathbf{c}_i, \tag{4}$$

This naive formulation, however, uses all Gaussians to render a pixel, which significantly hinders the rendering and training speed. In addition, such dense pixel-Gaussian correlation largely disrupts data locality, which is essential for fast random pixel access in GPU applications [Nystad et al. 2012; Vaidyanathan et al. 2023].

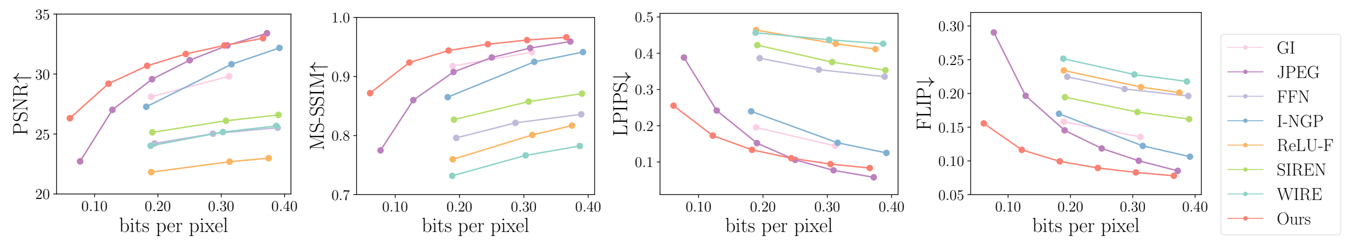


Fig. 3. *Rate-distortion curves (Section 4.3).* These results report the metric scores averaged over the evaluation set of 45 RGB images (Section 4.1).

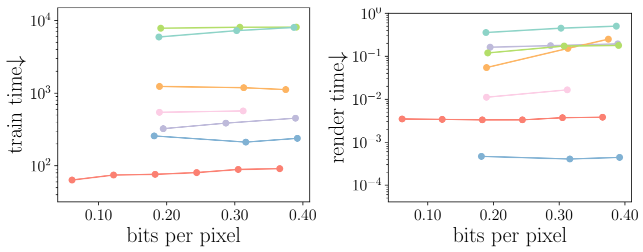


Fig. 4. *System performance (Section 4.4).* These results relate to the image experiments in Section 4.3 and share the same color scheme as Figure 3.

To resolve this issue, we take a tile-based rendering approach by following recent works [Lassner and Zollhofer 2021], and constrain the number of Gaussians that may contribute to a pixel. Specifically, we begin by subdividing the spatial support of an image into non-overlapping tiles of size $H_t \times W_t$. We then compute the minimum enclosing circle for each 2D Gaussian's 3-standard-deviation range (99.7% confidence interval) and check tile-circle intersection across all pairs to establish tile-Gaussian correspondence, where the set of Gaussians whose enclosing circles intersect tile T_j is denoted as $\mathcal{S}_j$, $1 \le j \le N_t$. After that, for each pixel $\mathbf{x} \in T_j$, we rank all Gaussians in $\mathcal{S}_j$ based on their values evaluated at $\mathbf{x}$ and only keep the top K. Finally, we normalize the values of the K remaining Gaussians, and aggregate their weighted colors to obtain $\mathbf{c}_r(\mathbf{x})$.

The final rendering equation for Image-GS is formulated as:

$$\mathbf{c}_r(\mathbf{x}) = \frac{1}{\sum_{i \in \mathcal{S}_j^K(\mathbf{x})} G_i(\mathbf{x})} \sum_{i \in \mathcal{S}_j^K(\mathbf{x})} G_i(\mathbf{x}) \cdot \mathbf{c}_i, \quad 1 \le j \le N_t \tag{5}$$

where $\mathcal{S}_j^K(\mathbf{x})$ denotes the set of top K Gaussians for $\mathbf{x} \in T_j$.

3.3 Content-Adaptive Initialization and Optimization

Building on the differentiable renderer introduced in Section 3.2, we optimize Gaussian attributes $\mathbf{p}_i$ through stochastic gradient descent to faithfully reconstruct any target image. Unlike the neural network models used in current neural image representations [Martel et al. 2021; Sitzmann et al. 2020; Tancik et al. 2020], Image-GS only consists of explicit features with physical meaning, and thus benefits from task-specific initialization for faster and higher-quality convergence. In particular, an ideal initialization should be guided by the image content, with the spatial distribution of Gaussians matching that of high-frequency image features.

To this end, we propose a content-adaptive position initialization strategy coupling image gradient guidance with uniform sampling. Specifically, during the position sampling of each Gaussian (we only sample pixel centers), the probability of a pixel $\mathbf{x}$ being sampled is a weighted sum of the relative magnitude of its local image gradient and a constant shared across all pixel locations. Notably, the former emphasizes image content-aware adaptivity, while the latter ensures adequate coverage of the entire image domain.

$$\mathbb{P}_{\text{init}}(\mathbf{x}) = \frac{(1-\lambda_{\text{init}}) \cdot \|\nabla I(\mathbf{x})\|_2}{\sum_{h=1}^{H}\sum_{w=1}^{W}\|\nabla I(\mathbf{x}_{h,w})\|_2} + \frac{\lambda_{\text{init}}}{H \cdot W}, \quad \lambda_{\text{init}} \in [0,1] \tag{6}$$

where H, W denote the height and width of the image, and $\nabla I(\cdot)$ is the image gradient operator. In addition, all Gaussians are assigned the target pixel color at their initialized location $\mathbf{c}_t(\mathbf{x})$.

During each optimization step, the set of Gaussians is rendered into an image using Equation (5), which is then used to compute a combination of L_1 and SSIM loss against the target image. Since the full pipeline is differentiable, all Gaussian attributes[1] can be updated via stochastic gradient descent to improve reconstruction.

Note that while the top-K ranking operation is non-differentiable, gradients do not flow through the operation itself but through the K retained Gaussians. Empirically (in Table 1), top-K normalization not only promotes data locality but also improves reconstruction quality. We hypothesize that the top-K normalization achieves this effect by acting as a form of regularization during optimization.

Besides the initially introduced Gaussians, we also progressively add new Gaussians to image areas exhibiting high reconstruction error. This is achieved by sampling pixels based on their error magnitude and initializing new Gaussians at sampled locations.

$$\mathbb{P}_{\text{add}}(\mathbf{x}) = \frac{|\mathbf{c}_r(\mathbf{x}) - \mathbf{c}_t(\mathbf{x})|}{\sum_{h=1}^{H}\sum_{w=1}^{W}|\mathbf{c}_r(\mathbf{x}_{h,w}) - \mathbf{c}_t(\mathbf{x}_{h,w})|}, \tag{7}$$

Figure 2 illustrates the optimization pipeline of Image-GS.

4 Evaluation

4.1 Experimental Setup

Dataset. Existing datasets for image compression evaluation, such as Kodak and CLIC, emphasize natural images and are mostly low-resolution (below 2 megapixel). In contrast, image assets used in the latest graphics workflows often have much higher resolutions and feature more diverse content, including stylized images. To address this gap, we collected 45 RGB images spanning 5 categories, vector-style, photograph, digital art, anime, and painting, with 9 samples per category. Additionally, we collected 19 texture stacks, each containing 9 channels (diffuse color, normal map, ambient occlusion,

[1]We empirically observed that optimizing the inverse of Gaussian scales $1/\mathbf{s}$ results in improved and faster convergence. Please refer to Section 4.1 for more details.

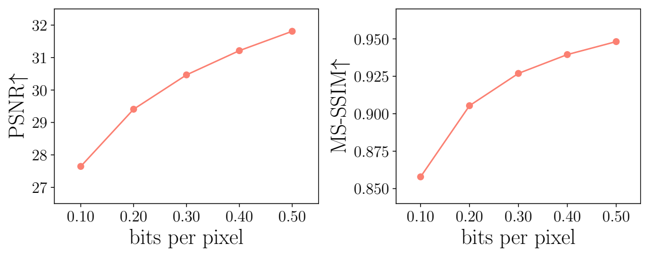


Fig. 5. *Rate-distortion curves on the CLIC2020 benchmark (Section 4.3).*

roughness, and metalness), for texture compression evaluation. All images and textures[2] have a resolution of 2K×2K.

Metrics. For image fidelity measurement, we use PSNR, MS-SSIM [Wang et al. 2003], LPIPS [Zhang et al. 2018], and FLIP [Andersson et al. 2020]. For experiments on textures, we omit LPIPS and FLIP, as they are specifically designed for natural images. We also report the model size (in kilobytes, KB) and bitrate (in bits per pixel, bpp, or bits per pixel per channel, bppc) to assess memory efficiency.

Implementation. Our differentiable rendering pipeline, built upon gsplat [Ye et al. 2024], is equipped with custom CUDA kernels for efficient forward and backward computation. Each tile of size 16×16 is processed by a separate CUDA block. The remaining components are implemented in PyTorch [Paszke et al. 2019]. The only trainable parameters in Image-GS, all quantized to float16, are the Gaussian attributes defined in Equation (3). By mapping the image domain to $[0,1]^2$, Image-GS supports target images of any aspect ratio and resolution. At initialization, all Gaussian positions $\boldsymbol{\mu}$ and colors $\mathbf{c}$ are populated via our content-adaptive sampling strategy (Equation (6)), while their scaling vectors $\mathbf{s}$ and rotation angles θ are initialized to 5 (pixels) and 0, respectively. Notably, instead of directly optimizing the raw Gaussian scales $\mathbf{s}$, which typically fall in the range $[5,10]$, we maintain and optimize their inverses $1/\mathbf{s}$ to improve convergence, since optimizing in the $[0,1]$ range yields smoother and more stable gradients. We use the Adam optimizer [Kingma and Ba 2015] to iteratively update these parameters against $L_1 + 0.1 \cdot L_{\text{SSIM}}$ for 5K steps. The learning rates for $(\boldsymbol{\mu}, \mathbf{c}, \mathbf{s}, \theta)$ are set to (5e−4, 5e−3, 2e−3, 2e−3) and remain constant throughout training. In our experiments, K and λ_{init} are set to 10 and 0.3, respectively. During training, we progressively allocate additional Gaussians to image regions exhibiting high fitting errors (Equation (7)). For a total budget of N_g Gaussians, training begins with $N_g/2$ Gaussians, and an additional $N_g/8$ are introduced every 0.5K steps until the budget is reached. Ablation studies on these design choices are provided in Table 1.

4.2 Visual Fidelity vs. Memory Efficiency

We assess Image-GS's rate-distortion performance on the evaluation set of 45 RGB images through error-guided progressive optimization (Section 3.3). As summarized in Figure 3, Image-GS achieves quality metrics of 32.99 ± 4.49 (PSNR), 0.966 ± 0.020 (MS-SSIM), 0.083 ± 0.057 (LPIPS), and 0.078 ± 0.029 (FLIP) at 0.366 bpp. Even at an ultra-low bitrate of 0.122 bpp, Image-GS maintains reasonable visual quality with metric scores of 29.20 ± 4.57 (PSNR), 0.924 ± 0.042 (MS-SSIM), 0.173 ± 0.082 (LPIPS), and 0.116 ± 0.043 (FLIP).

[2]Images and textures were sourced from Adobe Stock and Poly Haven, respectively.

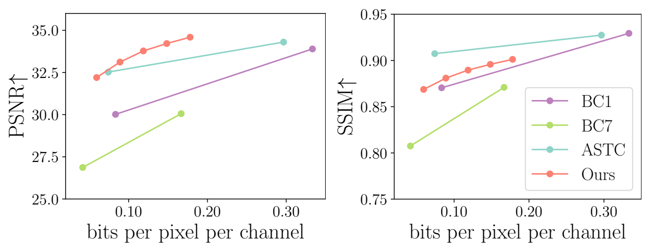


Fig. 6. *Rate-distortion curves on the 19 texture stacks (Section 4.5).*

Table 1. *Ablation studies on the design choices of Image-GS.*

Variants	PSNR↑	MS-SSIM↑	LPIPS↓
No color init	30.40 ± 4.73	0.951 ± 0.025	0.110 ± 0.064
No position init	29.88 ± 4.18	0.954 ± 0.028	0.135 ± 0.065
Random init	29.54 ± 4.15	0.948 ± 0.029	0.149 ± 0.067
No inverse scale	29.11 ± 4.58	0.933 ± 0.038	0.152 ± 0.073
No top-K norm	29.35 ± 4.43	0.944 ± 0.032	0.182 ± 0.086
Full	31.77 ± 4.73	0.960 ± 0.024	0.102 ± 0.062

Leveraging error-informed progressive optimization, Image-GS naturally constructs a smooth level-of-detail hierarchy in a single run without additional overhead. This design enables flexible quality control that adapts to the device capabilities at deployment. Besides, this design facilitates quality-driven compression, where Gaussians are progressively added until the required quality is met. In contrast, neural image representations based on implicit models or fixed data structures do not support straightforward control over compression quality. Figure 8 and Supplement B provide the results from several progressive runs that start at 0.061 bpp and terminate at 0.305 bpp.

4.3 Image Compression Performance

Baselines. We compare to 6 neural image representations: ReLU-F [Karnewar et al. 2022], SIREN [Sitzmann et al. 2020], FFN [Tancik et al. 2020], WIRE [Saragadam et al. 2023], I-NGP [Müller et al. 2022], and GI [Zhang et al. 2024]. We also include JPEG [Wallace 1991] as a conventional baseline for completeness. Since our objective is to represent high-resolution images at ultra-low bitrates, the allowable memory budget exceeds the range explored by most baselines. For fair comparisons, we adopt their official implementations and modify only the model sizes to match our target range. This is done by reducing the grid resolution (ReLU-F, I-NGP), the number of primitives (GI), and the number of layers and hidden dimensions (I-NGP, WIRE, SIREN, FFN). The evaluation set of 45 RGB images is employed for this experiment.

As shown in Figure 3, Image-GS outperforms all neural baselines across the entire bitrate range we evaluate. When the bitrate falls below 0.244, Image-GS even surpasses JPEG by a significant margin. Notably, both JPEG and GI leverage entropy coding, which breaks data locality, to improve performance, whereas Image-GS does not rely on such mechanisms. Figure 7 and Supplement A show several zoomed-in samples with error map overlays for visual comparison. Under this ultra-low bitrate regime, all baselines exhibit noticeable distortions and artifacts. For instance, decreasing the resolution of feature grids (ReLU-F, I-NGP) leads to block artifacts due to feature

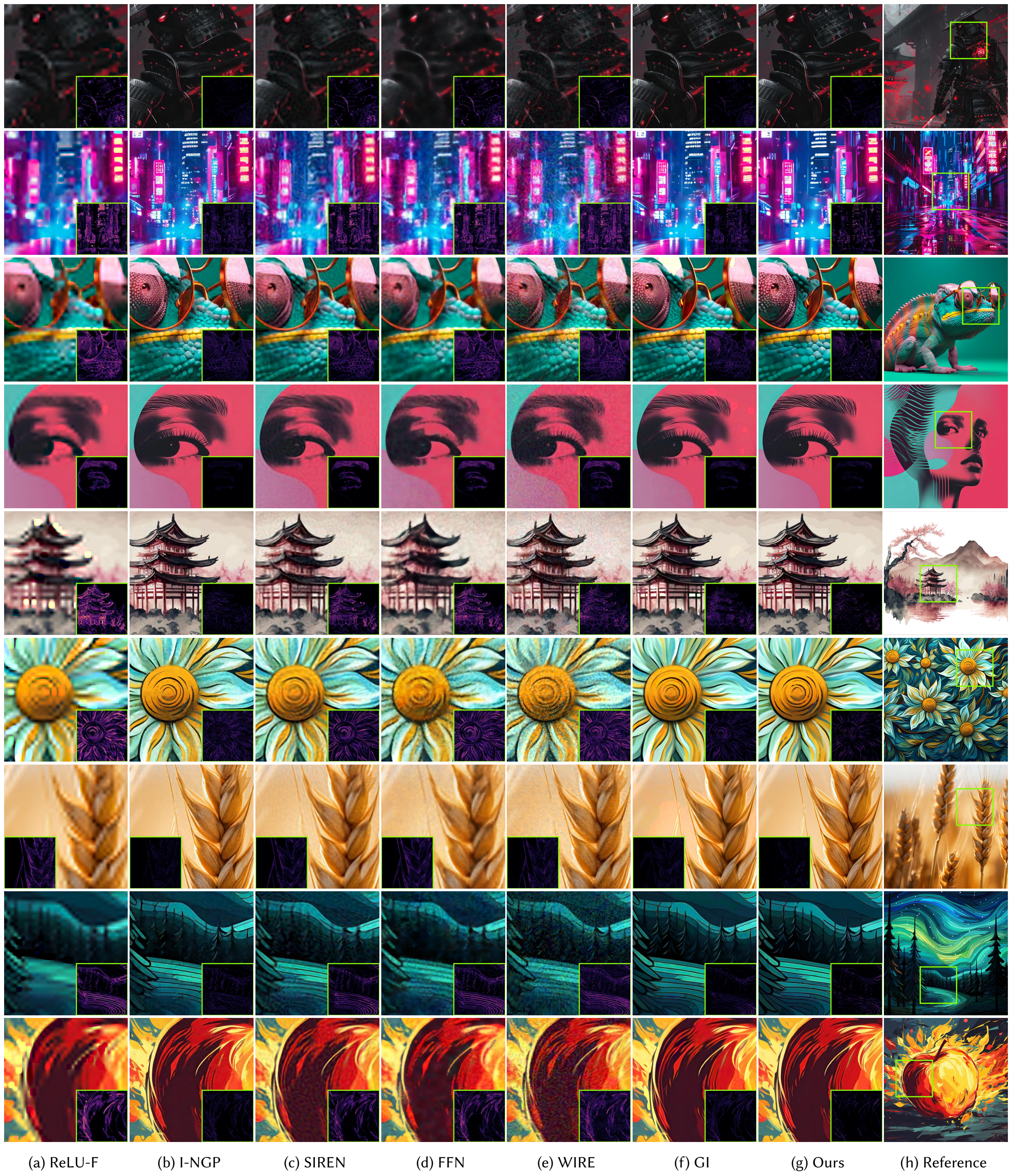


Fig. 7. *Qualitative comparison against conventional and neural image representations (Section 4.3).* For the 2K×2K-resolution results shown here, the model sizes (in KB) of ReLU-F, I-NGP, SIREN, FFN, WIRE, GI, and Image-GS are 164, 166, 161, 154, 159, 164, and 160, respectively.

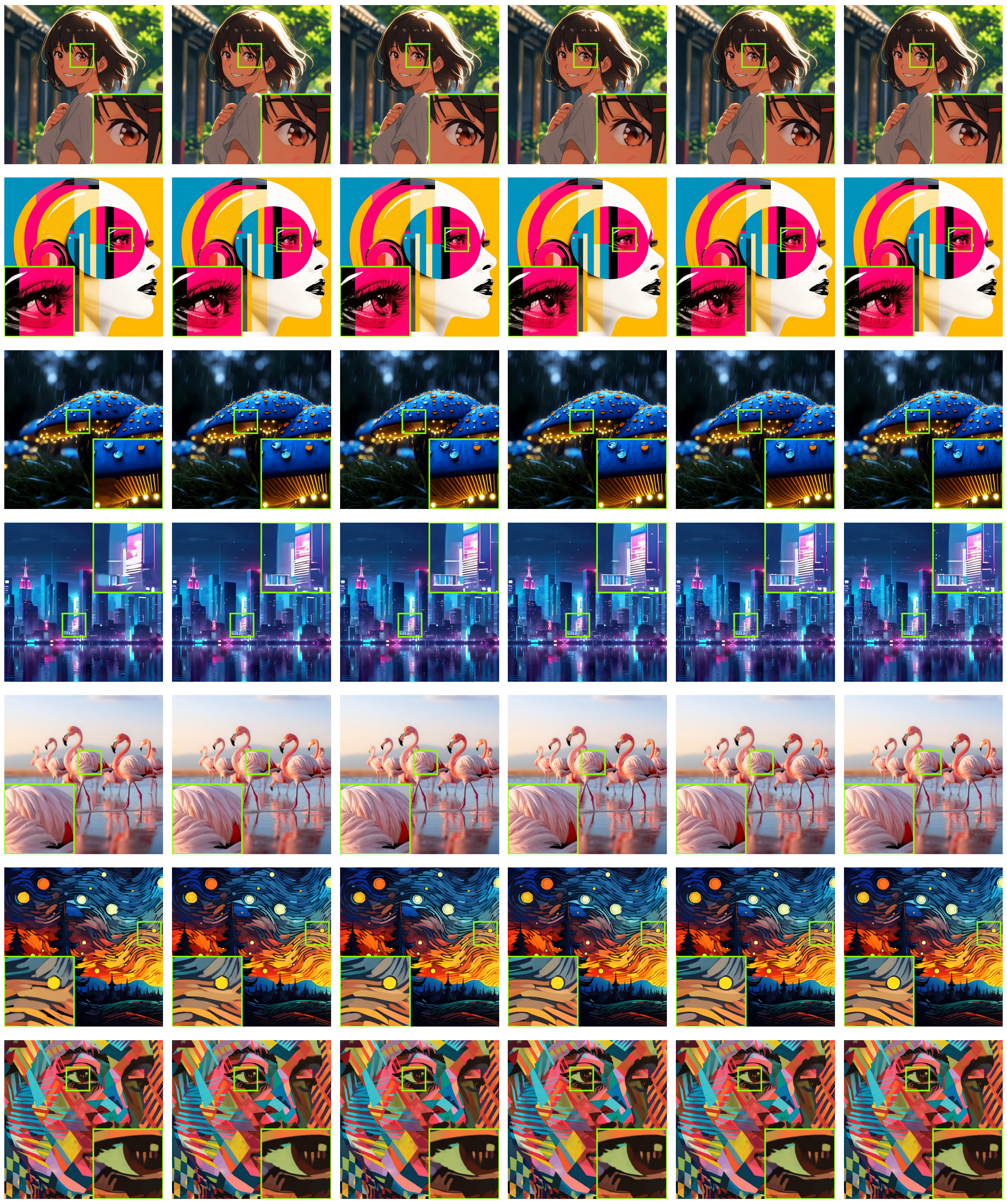

(a) Ours (0.061 bpp) (b) Ours (0.122 bpp) (c) Ours (0.183 bpp) (d) Ours (0.244 bpp) (e) Ours (0.305 bpp) (f) Reference

Fig. 8. *Image-GS's rate-distortion trade-off (Section 4.2).* Through error-guided progressive optimization (Section 3.3), Image-GS naturally constructs a smooth level-of-detail hierarchy in a single optimization run without additional overhead, enabling flexible quality adaptation to device capabilities.

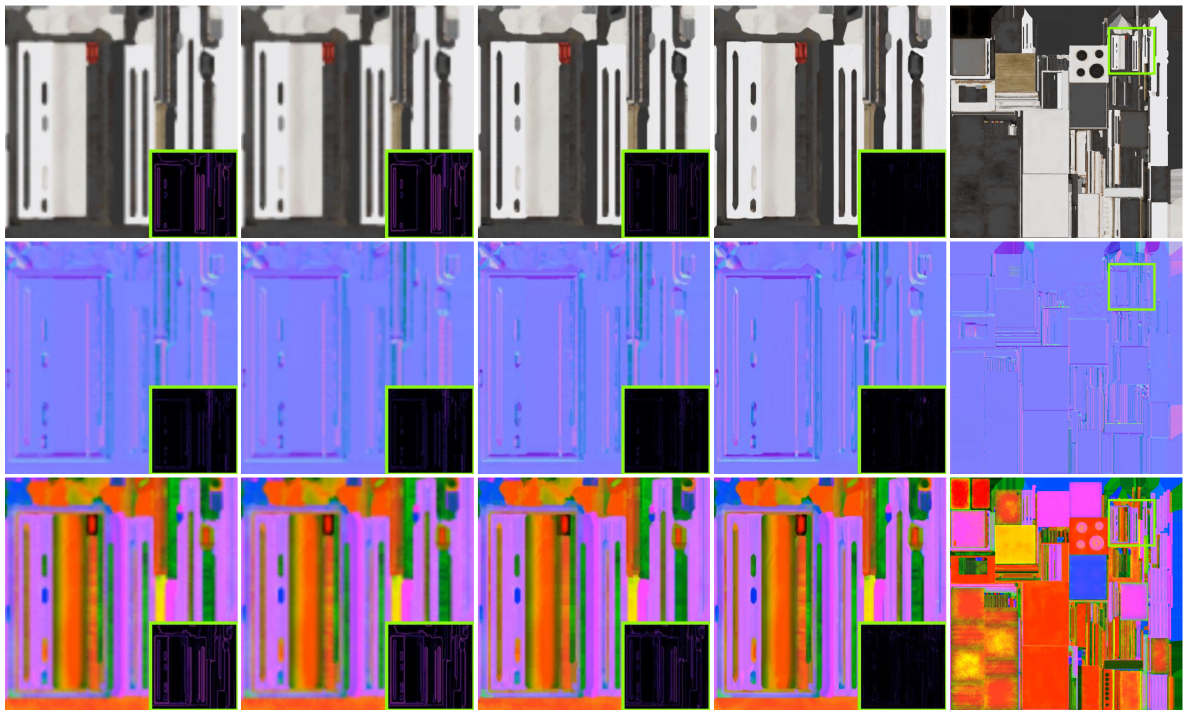


Fig. 9. *Qualitative comparison against industry-standard GPU texture compression algorithms (Section 4.5).*

vectors being interpolated at sparser locations. Implicit neural image representations (SIREN, FFN, WIRE) are prone to artifacts such as ringing, blurring, and ghosting, which become more pronounced after reducing the base network parameters.

We further ran Image-GS on the professional validation split of the CLIC2020 dataset [Toderici et al. 2020] to assess its compression performance on natural images. As illustrated in Figure 5, Image-GS achieves lower scores compared to the results for stylized images in Section 4.3. We attribute this performance drop to the more constrained Gaussian budgets on lower-resolution images at constant bitrates, as well as the prevalence of pixel-level camera sensor noise in natural images. As an explicit representation, Image-GS requires a sufficient number of Gaussian primitives to accurately capture spatially distinct image features. Supplement C shows several samples with zoomed-in overlays for visual comparison.

4.4 System Performance

Our efficient implementation enables fast training and rendering with Image-GS. For instance, optimizing 10K Gaussians for 1K steps at 2K×2K resolution takes an average of 18.74 seconds, while rendering (a single forward pass without gradient tracking) takes only 0.0037 seconds. This efficiency scales sub-linearly with the number of Gaussians: optimizing 50K Gaussians for 1K steps at 2K×2K resolution takes 26.32 seconds, and rendering takes 0.0045 seconds. All measurements were conducted on an Nvidia A6000 GPU.

As shown in Figure 4, Image-GS's rendering speed is second only to I-NGP. Thanks to designs such as inverse scale training and top-K normalization, Image-GS converges rapidly, typically within 3–4K steps (indicated by less than 0.1 PSNR and 0.001 SSIM improvements). It reaches 95% of its final performance in fewer than 400 steps and 99% within 2K steps. This fast convergence significantly reduces the overall training time for Image-GS. See the supplementary video for a real-time training demonstration at 8K×8K resolution.

4.5 Texture Compression Performance

Baselines. We compare with 3 industry-standard texture compression algorithms, BC1, BC7, and ASTC, using their implementations from NVIDIA Texture Tools[3]. Since they only support bitrates down to 4.0, 8.0, and 0.89 bpp, respectively, for RGB(A) textures, we extracted the compressed textures from their higher mipmap levels to match our target bitrate range for iso-bitrate comparison. These lower-resolution mipmaps were upsampled to $2K \times 2K$ resolution via bilinear interpolation before evaluation. We did not choose the alternative approach of running these baselines on downsampled versions of the target textures, as this would result in inconsistent reference across methods and make the comparison unfair.

As shown in Figure 6, Image-GS consistently outperforms BC1 and BC7 while achieving comparable performance to ASTC. Image-GS reaches quality metrics of 32.20 ± 4.06 (PSNR) and 0.869 ± 0.107 (SSIM) at an extreme bitrate of 0.059 bppc. Figure 9 and Supplement E show several zoomed-in texture stacks with error map overlays.

[3]https://developer.nvidia.com/texture-tools-exporter

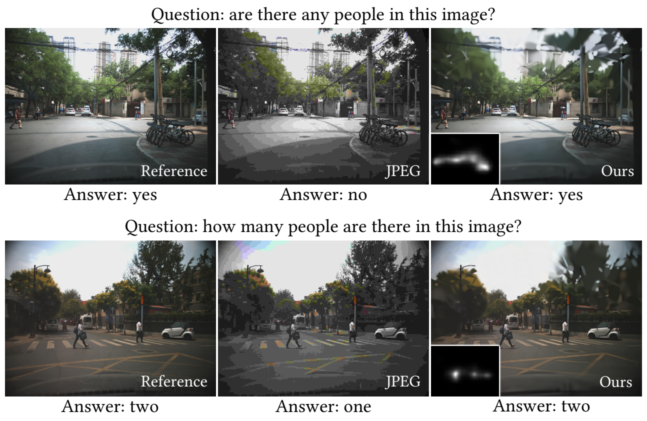


Fig. 10. *Semantics-aware compression (Section 5.1).* At an extreme rate of 0.2 bpp, Image-GS enables more accurate VQA results with BLIP-2 than JPEG.

5 Applications

5.1 Semantics-Aware Compression for Machine Vision

The exponential growth in the size of state-of-the-art machine vision models has been reshaping the AI deployment paradigm, with cloud provisioning increasingly supplanting local serving. In this context, the efficient storage and transfer of visual content, while preserving the information essential to the underlying applications, are critical to both the end users and cloud service providers [Hu et al. 2021].

Thanks to the explicit nature of Image-GS, we can readily factor in the distribution of such information through visual saliency analysis [Itti et al. 1998] and accordingly distribute Gaussian primitives over the image domain to better preserve the important semantic content therein. Specifically, given a target image, we first take an off-the-shelf saliency predictor [Jia and Bruce 2020] to extract its saliency map $S \in \mathbb{R}_{+}^{H \times W}$, then perform saliency-guided Gaussian position initialization with sampling probability:

$$\mathbb{P}_{\text{init}}(\mathbf{x}) = \frac{(1-\lambda_{\text{init}}) \cdot S(\mathbf{x})}{\sum_{h=1}^{H}\sum_{w=1}^{W} S(\mathbf{x}_{h,w})} + \frac{\lambda_{\text{init}}}{H \cdot W} \quad \lambda_{\text{init}} \in [0,1] \tag{8}$$

$\lambda_{\text{init}} = 0.1$ balances saliency guidance and uniform coverage.

We demonstrate the semantics-aware compression performance of Image-GS on a vision-language task: visual question answering (VQA) [Antol et al. 2015]. JPEG was used as the baseline and BLIP-2 [Li et al. 2023] was used to generate responses. We experimented with 20 randomly sampled images from the TJU-DHD dataset [Pang et al. 2020] and prepared custom questions for evaluation.

Figure 10 visualizes the results of 2 image-question pairs. At an extreme bitrate of 0.2 bpp, Image-GS more effectively preserves task-relevant semantic image features during compression and enables outputs that better align with the uncompressed counterpart. Out of the 20 image-question pairs, Image-GS achieved the same response as the reference 12 times, while JPEG only achieved 4 times. These results demonstrate Image-GS's potential for serving as a compact yet robust encoding of visual inputs for machine vision applications.

5.2 Joint Image Compression and Restoration

The low-pass nature of Gaussian functions endows Image-GS with remarkable robustness against a range of common image distortions and artifacts at low bitrates, including the ringing and contouring patterns introduced by lossy compression [Vander Kam et al. 1999], the color banding and aliasing caused by quantization errors [Lorre and Gillespie 1980], and the various forms of noise that originate from transmission or imaging processes [Wei et al. 2020].

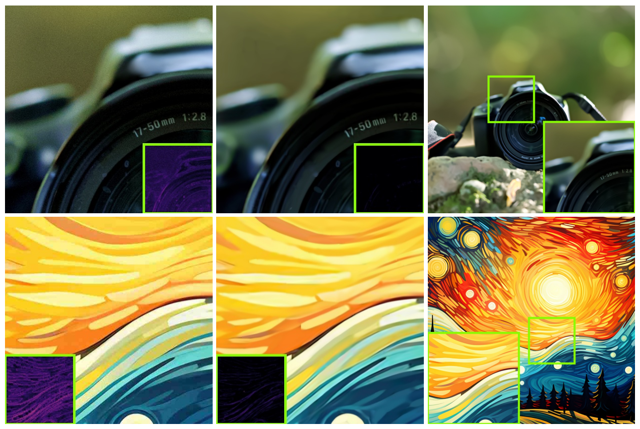


Fig. 11. *Joint image compression and restoration (Section 5.2).* At low bitrates, Image-GS effectively removes high-frequency artifacts from the input images while preserving detailed semantic content therein.

We demonstrate that Image-GS effectively achieves joint image compression and restoration at low bitrates. We experimented with 2 sets of 2K×2K images, where 6 were JPEG-compressed (average size is 187 KB) and 8 were photos containing noticeable sensor noise. For quantitative evaluation, the corresponding uncompressed PNGs were used as ground truth for the 6 JPEG images, while AI-denoised outputs served as ground truth for the 8 photos.

As shown in Figure 11, Image-GS eliminates most artifacts and noise from the input images while preserving the detailed content therein. This is because the limited representation budget (160 KB, 78.64× compression) forces Image-GS to prioritize bits on the more prominent image content instead of inconsistent pixel-level noise. Despite high compression ratios, images compressed by Image-GS demonstrate consistently improved fidelity to the ground truth. On the 8 noise-corrupted photos, it achieves average gains of 1.782 in PSNR and 0.011 in MS-SSIM. On the 6 JPEG-compressed images, it achieves average gains of 0.354 in PSNR and 0.012 in MS-SSIM. Notably, these restoration effects naturally emerge from Image-GS's compression process, requiring no additional post-processing. More results can be found in Supplement D.

6 Limitations and Discussion

Spatially adaptive optimization. Although Image-GS is designed to be content-adaptive, its current optimization pipeline prioritizes large image features and struggles with reconstructing images that are rich in pixel-level details, such as natural images (see examples in Supplement C). Inspired by hybrid representations with spatially adaptive optimization [Martel et al. 2021], we plan to incorporate a dynamic binary space partitioning tree to guide the spatial distribution of Gaussians and adaptively scale the gradients they receive, with the tree structure and Gaussian attributes jointly updated during optimization. This approach encourages comparable importance across image features of varying scales.

Dynamic content. We demonstrated in this research that images can be efficiently represented using an explicit basis of colored 2D Gaussians. Following recent works that incorporate dynamics into Gaussian-based scene representations [Diolatzis et al. 2024; Luiten et al. 2024], we plan to extend Image-GS to efficient video representations by modeling the motion of 2D Gaussians in the image plane. We envision this extension benefiting graphics applications such as panoramic video streaming in extended reality.

7 Conclusion

In this work, we proposed Image-GS, an explicit image representation based on anisotropic, colored 2D Gaussians. Image-GS supports favorable rate-distortion trade-offs, hardware-friendly fast random access, and flexible quality controls through a smooth level-of-detail stack. Its content-adaptive design effectively captures non-uniform image features and preserves fine details under constrained memory budgets. We hope this research inspires future advances in developing novel representations of visual data.

Acknowledgments

This research is partially supported by the NSF grants #2232817 and #2225861, and an Intel-sponsored research program.

References

2024. Texture Block Compression in Direct3D 11. https://learn.microsoft.com/en-us/windows/win32/direct3d11/texture-block-compression-in-direct3d-11.

Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. 2019. *Real-time rendering.* AK Peters/crc Press.

Jyrki Alakuijala, Ruud Van Asseldonk, Sami Boukortt, Martin Bruse, Iulia-Maria Comșa, Moritz Firsching, Thomas Fischbacher, Evgenii Kliuchnikov, Sebastian Gomez, Robert Obryk, et al. 2019. JPEG XL next-generation image compression architecture and coding tools. In *Applications of digital image processing XLII*, Vol. 11137. SPIE, 112–124.

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D Fairchild. 2020. FLIP: A Difference Evaluator for Alternating Images. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 2 (2020), 15–1.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision.* 2425–2433.

Marc Antonini, Michel Barlaud, Pierre Mathieu, and Ingrid Daubechies. 1992. Image coding using wavelet transform. *IEEE Trans. Image Processing* 1 (1992), 20–5.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. 2017. End-to-end optimized image compression. In *5th International Conference on Learning Representations, ICLR 2017.*

Zhihua Ban, Jianguo Liu, and Li Cao. 2018. Superpixel segmentation using Gaussian mixture model. *IEEE Transactions on Image Processing* 27, 8 (2018), 4105–4117.

Yash Belhe, Michaël Gharbi, Matthew Fisher, Iliyan Georgiev, Ravi Ramamoorthi, and Tzu-Mao Li. 2023. Discontinuity-Aware 2D Neural Fields. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–11.

Graham Campbell, Thomas A DeFanti, Jeff Frederiksen, Stephen A Joyce, Lawrence A Leske, John A Lindberg, and Daniel J Sandin. 1986. Two bit/pixel full color encoding. *ACM SIGGRAPH Computer Graphics* 20, 4 (1986), 215–223.

Turgay Celik and Tardi Tjahjadi. 2011. Automatic image equalization and contrast enhancement using Gaussian mixture modeling. *IEEE transactions on image processing* 21, 1 (2011), 145–156.

Yinbo Chen, Sifei Liu, and Xiaolong Wang. 2021. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 8628–8638.

Yue Chen, Debargha Murherjee, Jingning Han, Adrian Grange, Yaowu Xu, Zoe Liu, Sarah Parker, Cheng Chen, Hui Su, Urvang Joshi, et al. 2018. An overview of core coding tools in the AV1 video codec. In *2018 picture coding symposium (PCS).* IEEE, 41–45.

Zhang Chen, Zhong Li, Liangchen Song, Lele Chen, Jingyi Yu, Junsong Yuan, and Yi Xu. 2023. Neurbf: A neural fields representation with adaptive radial basis functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 4182–4194.

Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. 2020. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 7939–7948.

Edward Delp and O Mitchell. 1979. Image compression using block truncation coding. *IEEE transactions on Communications* 27, 9 (1979), 1335–1342.

Stavros Diolatzis, Tobias Zirr, Alexander Kuznetsov, Georgios Kopanas, and Anton Kaplanyan. 2024. N-dimensional gaussians for fitting of high dimensional functions. In *ACM SIGGRAPH 2024 Conference Papers.* 1–11.

Emilien Dupont, Adam Golinski, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. 2021. COIN: COmpression with Implicit Neural representations. In *Neural Compression: From Information Theory to Applications–Workshop@ ICLR 2021.*

E Dupont, H Loya, M Alizadeh, A Golinski, YW Teh, and A Doucet. 2022. COIN++: neural compression across modalities. *Transactions on Machine Learning Research* 2022, 11 (2022).

Ziv Epstein, Aaron Hertzmann, Investigators of Human Creativity, Memo Akten, Hany Farid, Jessica Fjeld, Morgan R Frank, Matthew Groh, Laura Herman, Neil Leach, et al. 2023. Art and the science of generative AI. *Science* 380, 6650 (2023), 1110–1111.

Ben Fei, Jingyi Xu, Rui Zhang, Qingyuan Zhou, Weidong Yang, and Ying He. 2024. 3d gaussian splatting as new era: A survey. *IEEE Transactions on Visualization and Computer Graphics* (2024).

Yueyu Hu, Wenhan Yang, Zhan Ma, and Jiaying Liu. 2021. Learning end-to-end lossy image compression: A benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 8 (2021), 4194–4211.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers.* 1–11.

Laurent Itti, Christof Koch, and Ernst Niebur. 1998. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence* 20, 11 (1998), 1254–1259.

Sen Jia and Neil DB Bruce. 2020. Eml-net: An expandable multi-layer network for saliency prediction. *Image and vision computing* 95 (2020), 103887.

Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. 2022. Relu fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 conference proceedings.* 1–9.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–14.

Hyunjik Kim, Matthias Bauer, Lucas Theis, Jonathan Richard Schwarz, and Emilien Dupont. 2024. C3: High-performance and low-complexity neural compression from a single image or video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 9347–9358.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.*

Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. 2021. Point-Based Neural Rendering with Per-View Optimization. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 29–43.

Théo Ladune, Pierrick Philippe, Félix Henry, Gordon Clare, and Thomas Leguay. 2023. Cool-chic: Coordinate-based low complexity hierarchical image codec. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 13515–13522.

Christoph Lassner and Michael Zollhofer. 2021. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 1440–1449.

Thomas Leguay, Théo Ladune, Pierrick Philippe, Gordon Clare, Félix Henry, and Olivier Déforges. 2023. Low-complexity overfitted neural image codec. In *2023 IEEE 25th International Workshop on Multimedia Signal Processing (MMSP).* IEEE, 1–6.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning.* PMLR, 19730–19742.

Jean J Lorre and Alan R Gillespie. 1980. Artifacts in digital images. In *Applications of Digital Image Processing to Astronomy*, Vol. 264. SPIE, 123–135.

Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. 2024. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV).* IEEE, 800–809.

Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. 2021. Acorn: adaptive coordinate networks for neural scene representation. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–13.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15.

Milad Niknejad, Hossein Rabbani, and Massoud Babaie-Zadeh. 2015. Image restoration using Gaussian mixture models with spatially constrained patch clustering. *IEEE*

Transactions on Image Processing 24, 11 (2015), 3624–3636.

Jörn Nystad, Anders Lassen, Andy Pomianowski, Sean Ellis, and Tom Olson. 2012. Adaptive scalable texture compression. In *Proceedings of the Fourth ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics*. 105–114.

Yanwei Pang, Jiale Cao, Yazhao Li, Jin Xie, Hanqing Sun, and Jinfeng Gong. 2020. TJU-DHD: A diverse high-resolution dataset for object detection. *IEEE Transactions on Image Processing* 30 (2020), 207–219.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.

Ryan Po, Wang Yifan, Vladislav Golyanik, Kfir Aberman, Jonathan T Barron, Amit Bermano, Eric Chan, Tali Dekel, Aleksander Holynski, Angjoo Kanazawa, et al. 2024. State of the art on diffusion models for visual computing. In *Computer Graphics Forum*, Vol. 43. Wiley Online Library, e15063.

Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. 2023. Wire: Wavelet implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18507–18516.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. 2020. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems* 33 (2020), 7462–7473.

Jacob Ström and Tomas Akenine-Möller. 2005. i PACKMAN: High-quality, low-complexity texture compression for mobile phones. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. 63–70.

Jacob Ström and Martin Pettersson. 2007. ETC 2: texture compression using invalid combinations. In *Graphics Hardware*, Vol. 7. 49–54.

Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 2024. 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20675–20685.

Jianjun Sun, Yan Zhao, Shigang Wang, and Jian Wei. 2021. Image compression based on Gaussian mixture model constrained using Markov random field. *Signal Processing* 183 (2021), 107990.

Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. 2022. Variable bitrate neural fields. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–9.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems* 33 (2020), 7537–7547.

Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. 2017. Lossy Image Compression with Compressive Autoencoders. In *International Conference on Learning Representations*.

George Toderici, Wenzhe Shi, Radu Timofte, Lucas Theis, Johannes Balle, Eirikur Agustsson, Nick Johnston, and Fabian Mentzer. 2020. Workshop and challenge on learned image compression (clic2020). In *CVPR*.

Peihan Tu, Li-Yi Wei, and Matthias Zwicker. 2024. Compositional Neural Textures. In *SIGGRAPH Asia 2024 Conference Papers*. 1–11.

Karthik Vaidyanathan, Marco Salvi, Bartlomiej Wronski, Tomas Akenine-Moller, Pontus Ebelin, and Aaron Lefohn. 2023. Random-Access Neural Compression of Material Textures. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–25.

Rick A Vander Kam, Ping Wah Wong, and Robert M Gray. 1999. JPEG-compliant perceptual coding for a grayscale image printing pipeline. *IEEE transactions on image processing* 8, 1 (1999), 1–14.

Gregory K Wallace. 1991. The JPEG still picture compression standard. *Commun. ACM* 34, 4 (1991), 30–44.

Zhou Wang, Eero P Simoncelli, and Alan C Bovik. 2003. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Vol. 2. Ieee, 1398–1402.

Kaixuan Wei, Ying Fu, Jiaolong Yang, and Hua Huang. 2020. A physics-based noise formation model for extreme low-light raw denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2758–2767.

Terry A Welch. 1985. High speed data compression and decompression apparatus and method. US Patent 4,558,302.

Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, et al. 2024. gsplat: An open-source library for Gaussian splatting. *arXiv preprint arXiv:2409.06765* (2024).

Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. 2019. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.

Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Yan Wang, Hongwei Qin, Guo Lu, Jing Geng, and Jun Zhang. 2024. Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting. In *European Conference on Computer Vision*. Springer, 327–345.

Xiaosu Zhu, Jingkuan Song, Lianli Gao, Feng Zheng, and Heng Tao Shen. 2022. Unified multivariate gaussian mixture for efficient neural image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 17612–17621.

A Additional Image Compression Results

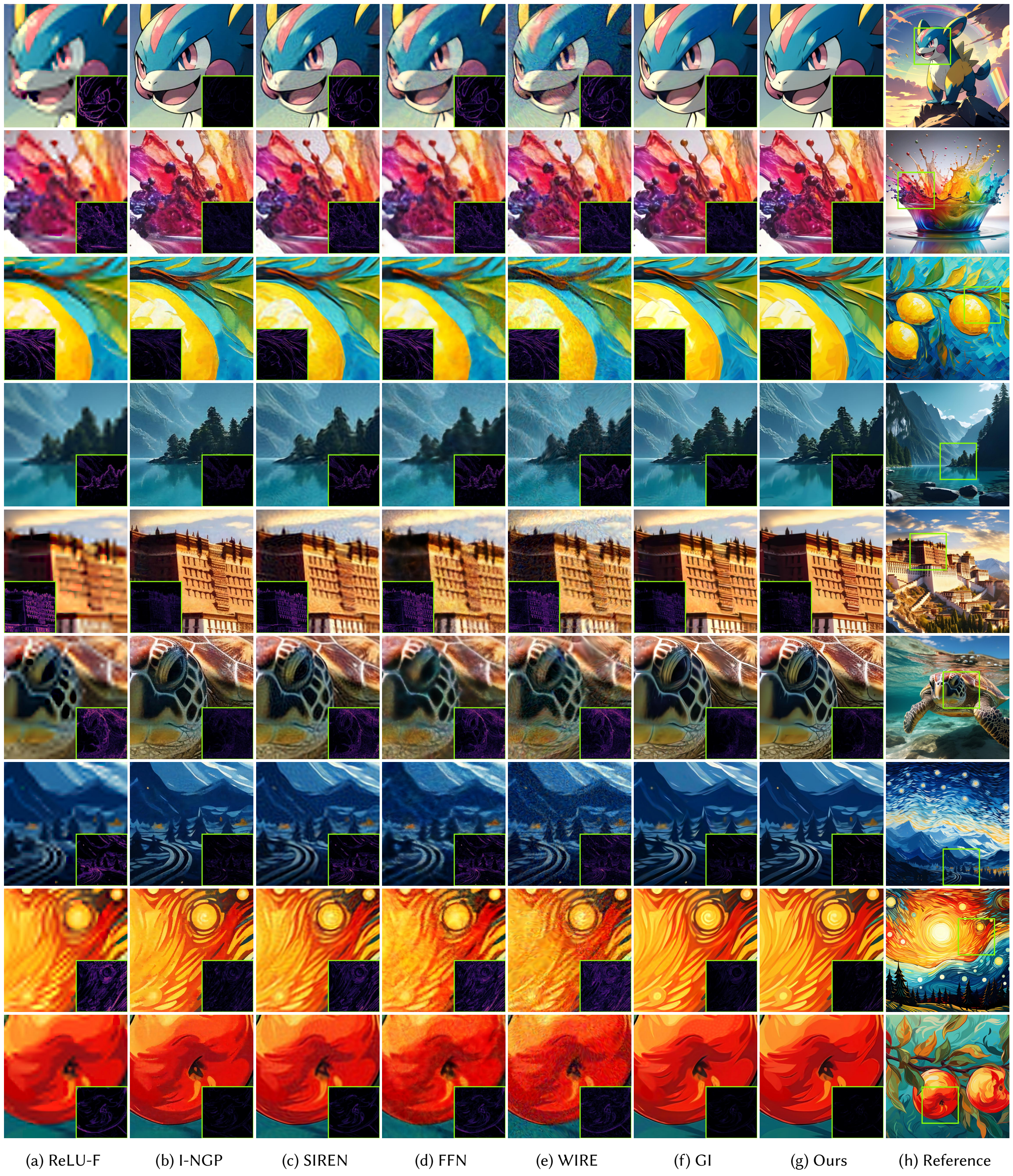


Fig. 12. *Qualitative comparison against conventional and neural image representations (Section 4.3).* For the 2K×2K-resolution results shown here, the model sizes (in KB) of ReLU-F, I-NGP, SIREN, FFN, WIRE, GI, and Image-GS are 164, 166, 161, 154, 159, 164, and 160, respectively.

B Additional Rate-Distortion Trade-off Results

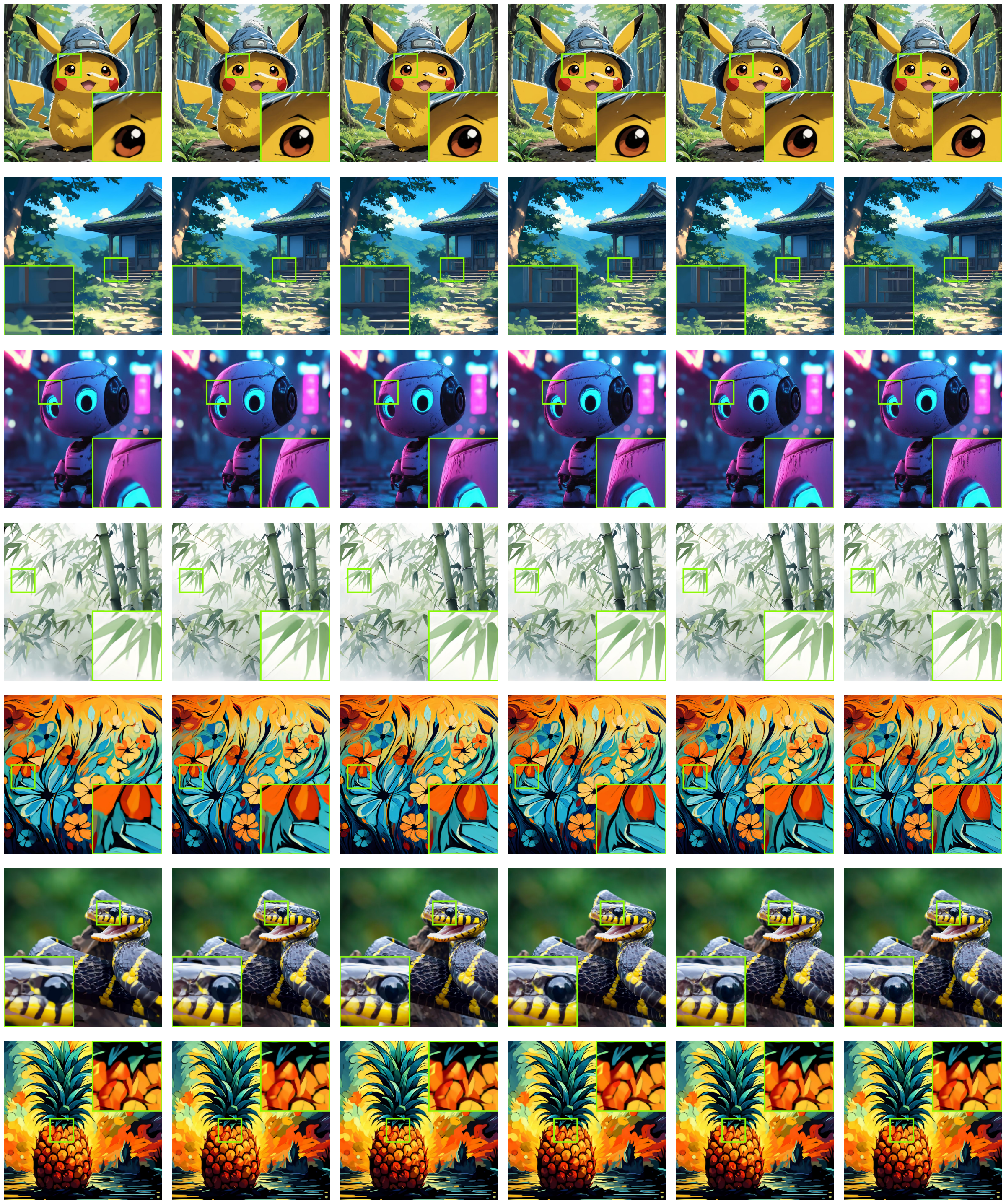
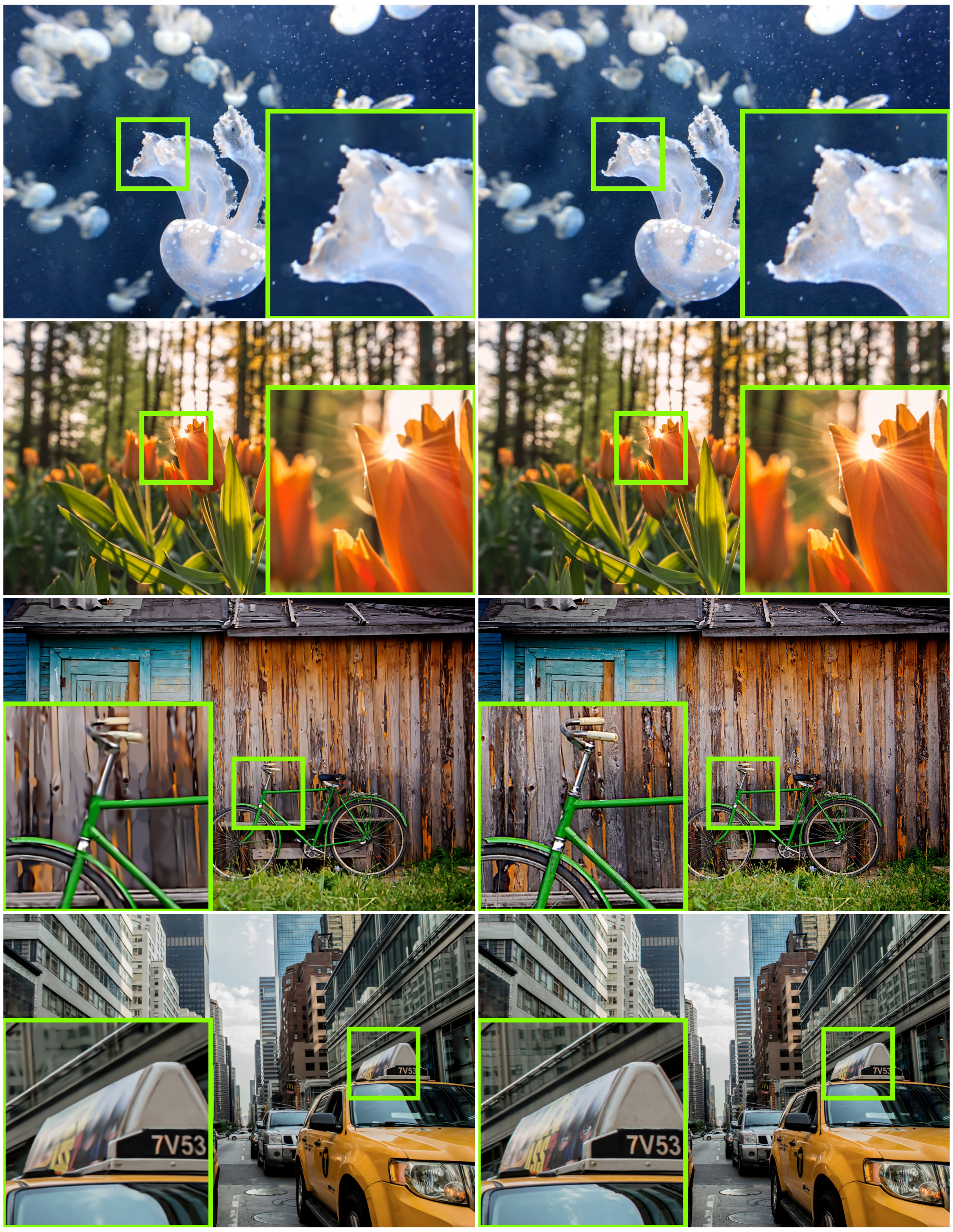


Fig. 13. *Image-GS's rate-distortion trade-off (Section 4.2).* Through error-guided progressive optimization (Section 3.3), Image-GS naturally constructs a smooth level-of-detail hierarchy in a single optimization run without additional overhead, enabling flexible quality adaptation to device capabilities.

C Additional Image Compression Results on The CLIC2020 Benchmark



(a) Ours (0.50 bpp) (b) Reference

Fig. 14. *Qualitative results on the professional validation split of the CLIC2020 dataset [Toderici et al. 2020] (Section 4.3).*

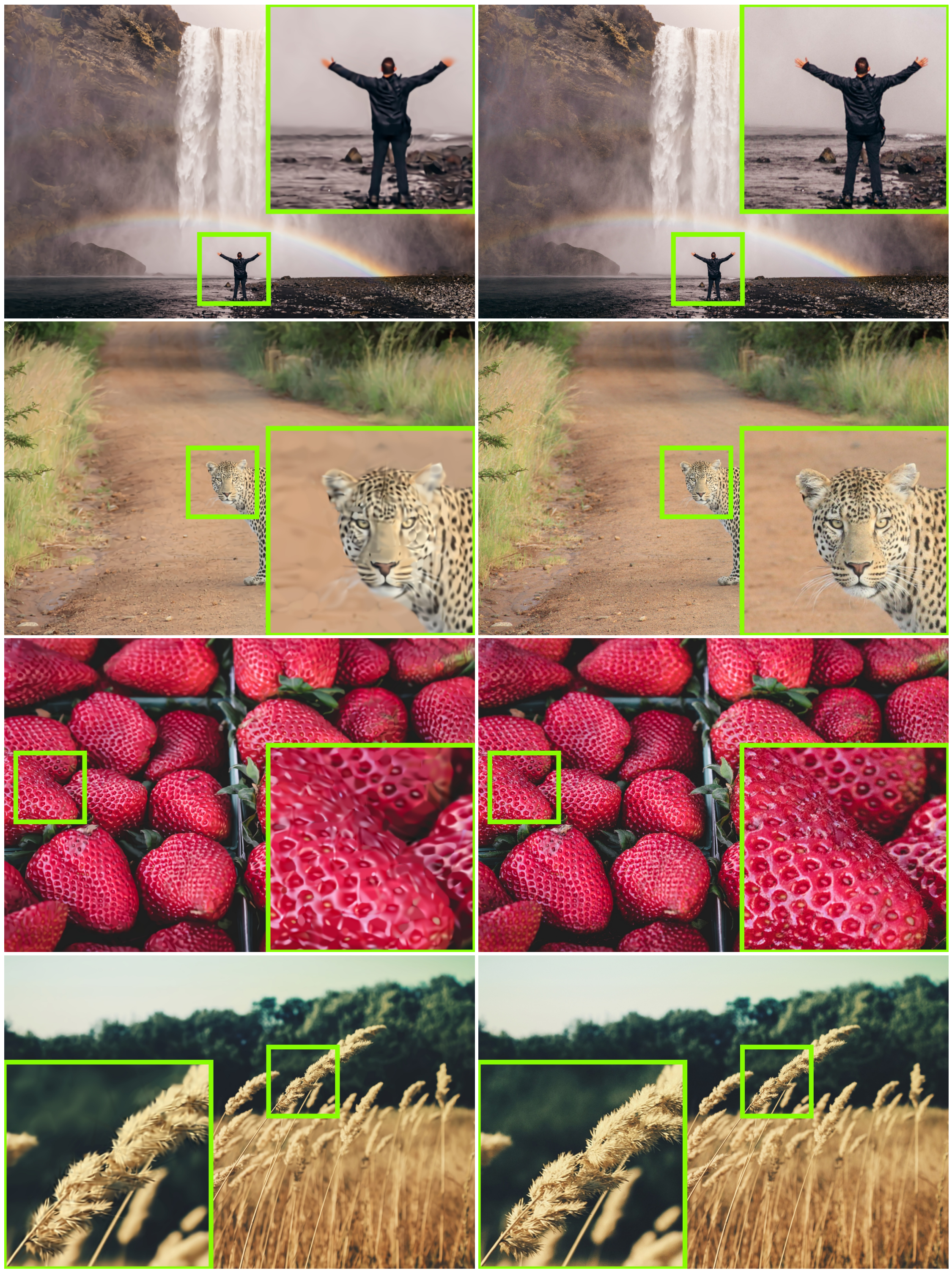


Fig. 15. *Qualitative results on the professional validation split of the CLIC2020 dataset [Toderici et al. 2020] (Section 4.3).*

D Additional Joint Image Compression and Restoration Results

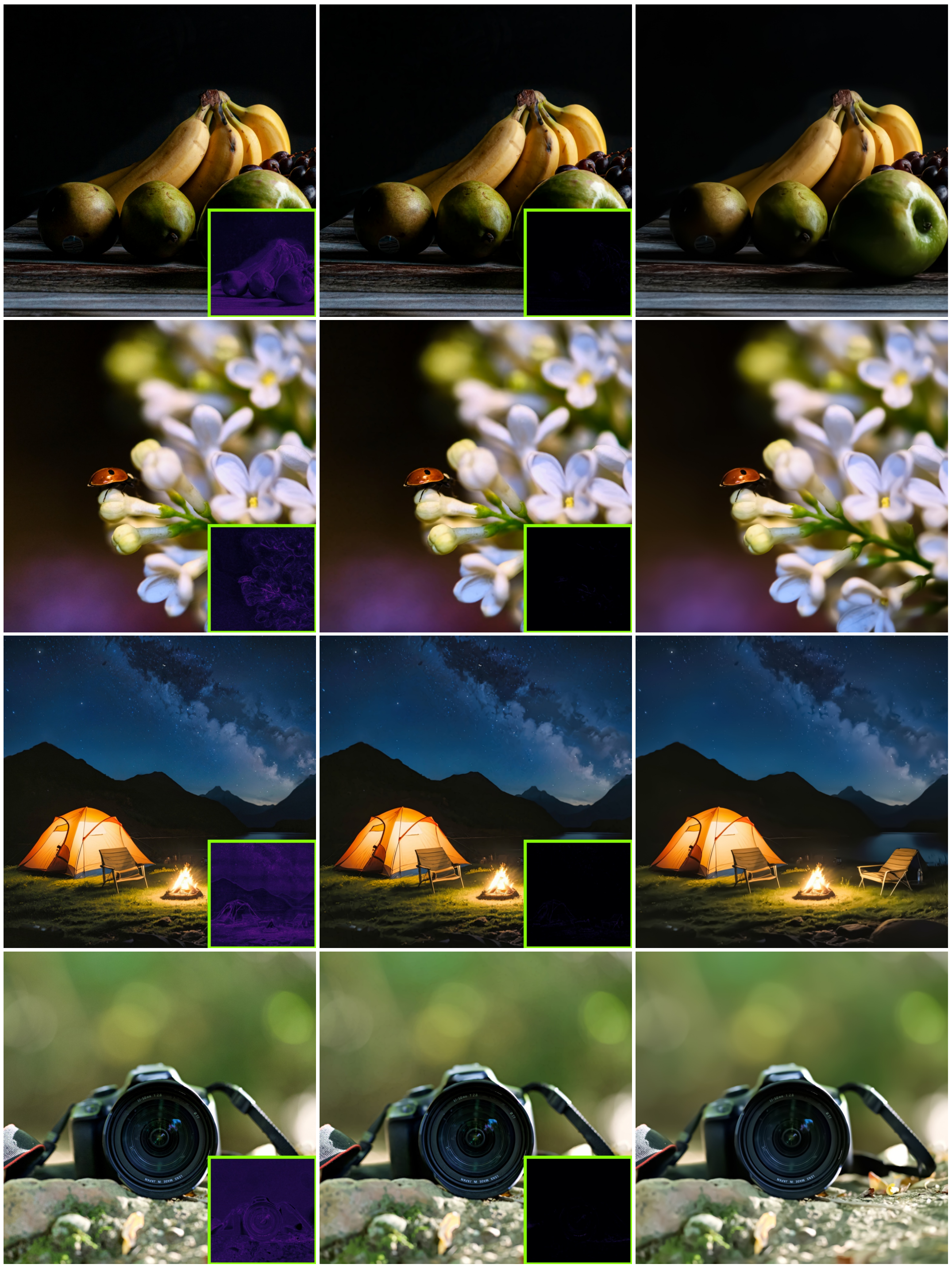

(a) Image with camera sensor noise
(b) Image restored by Image-GS
(c) AI-denoised reference image

Fig. 16. *Joint image compression and restoration (Section 5.2).* At low bitrates, Image-GS effectively removes high-frequency artifacts from the input images while preserving detailed semantic content therein.

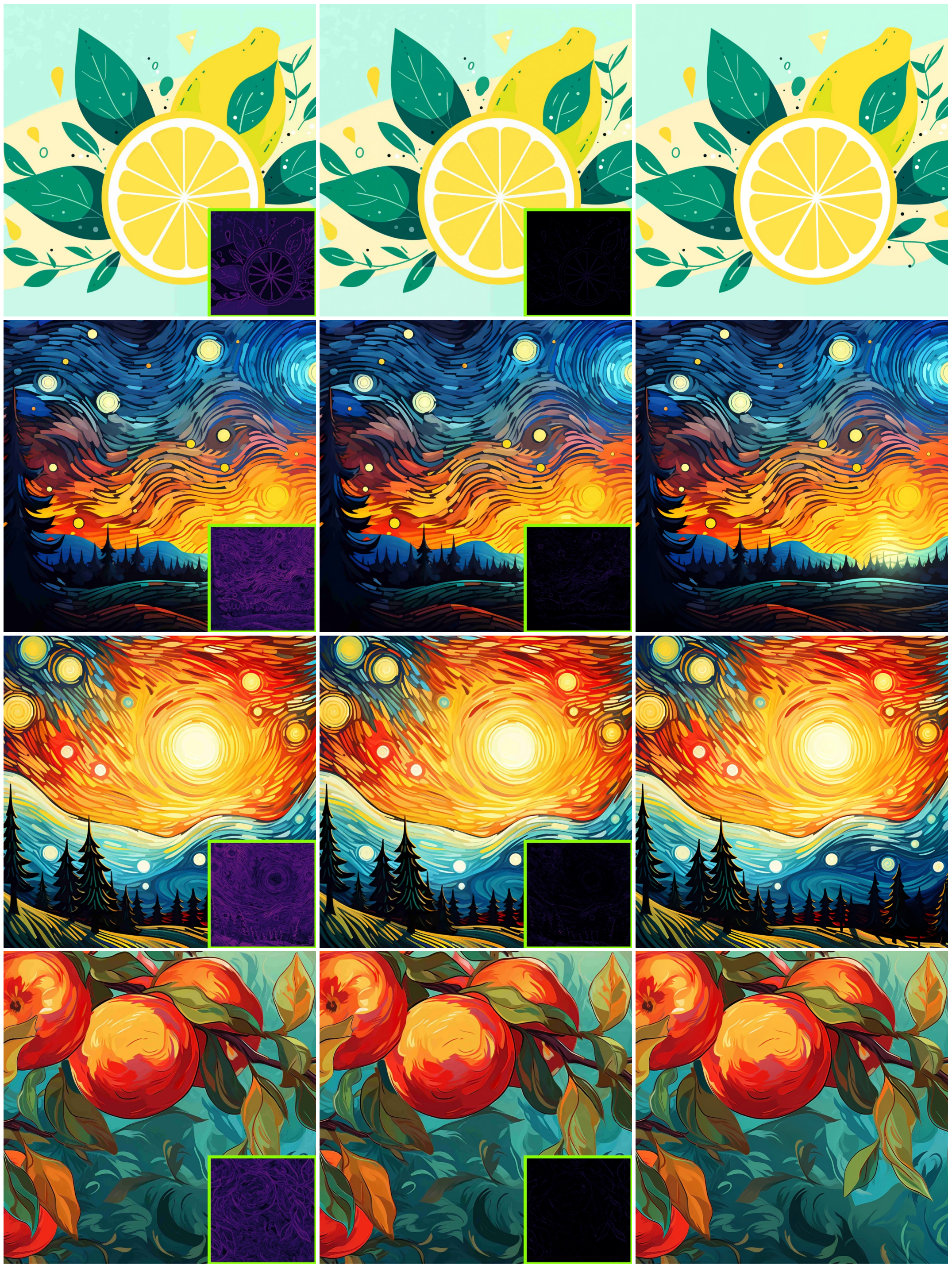

(a) Image compressed by JPEG
(b) Image restored by Image-GS
(c) Uncompressed reference image

Fig. 17. *Joint image compression and restoration (Section 5.2).* At low bitrates, Image-GS effectively removes high-frequency artifacts from the input images while preserving detailed semantic content therein.

E Additional Texture Compression Results.

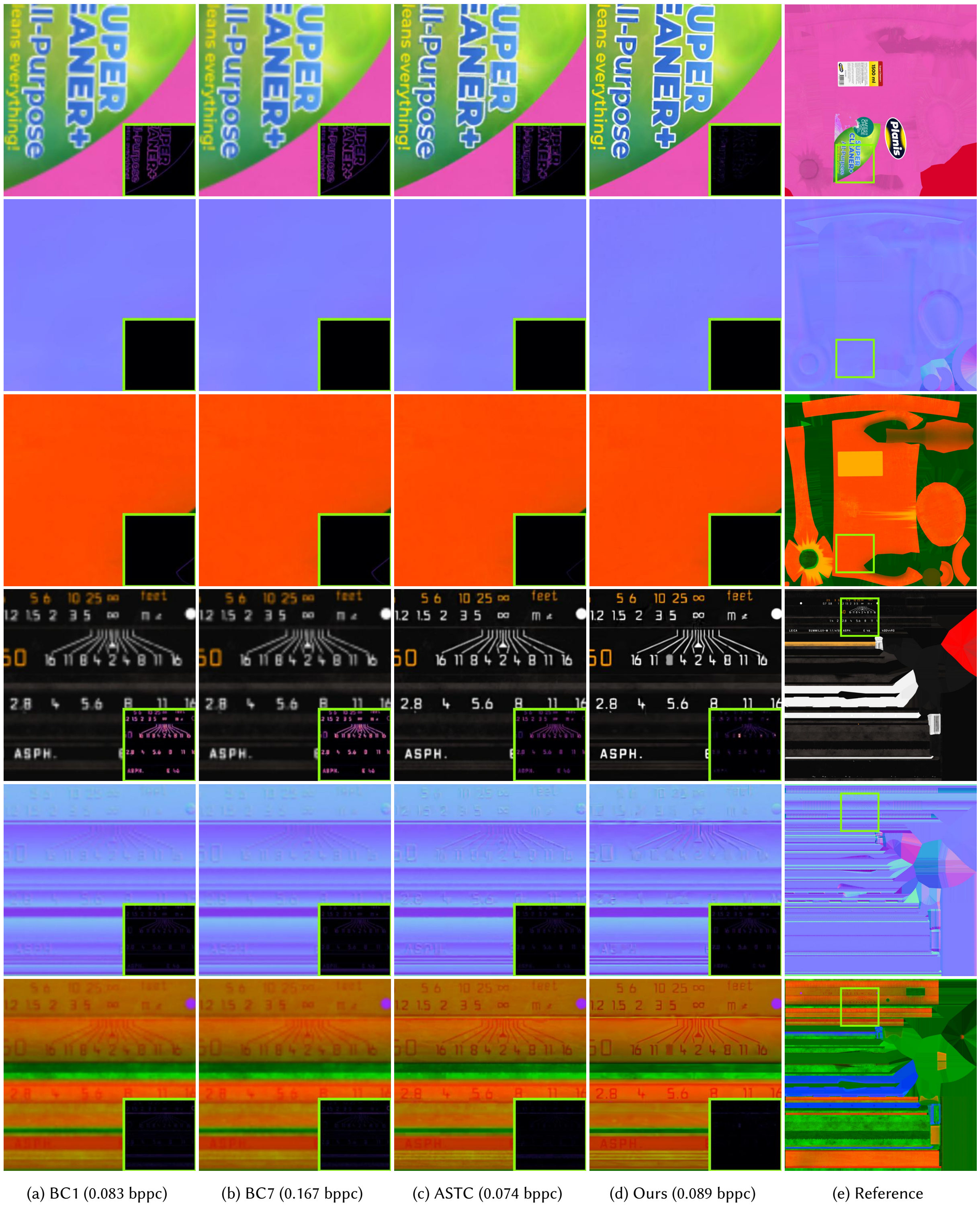


Fig. 18. *Qualitative comparison against industry-standard GPU texture compression algorithms (Section 4.5).*

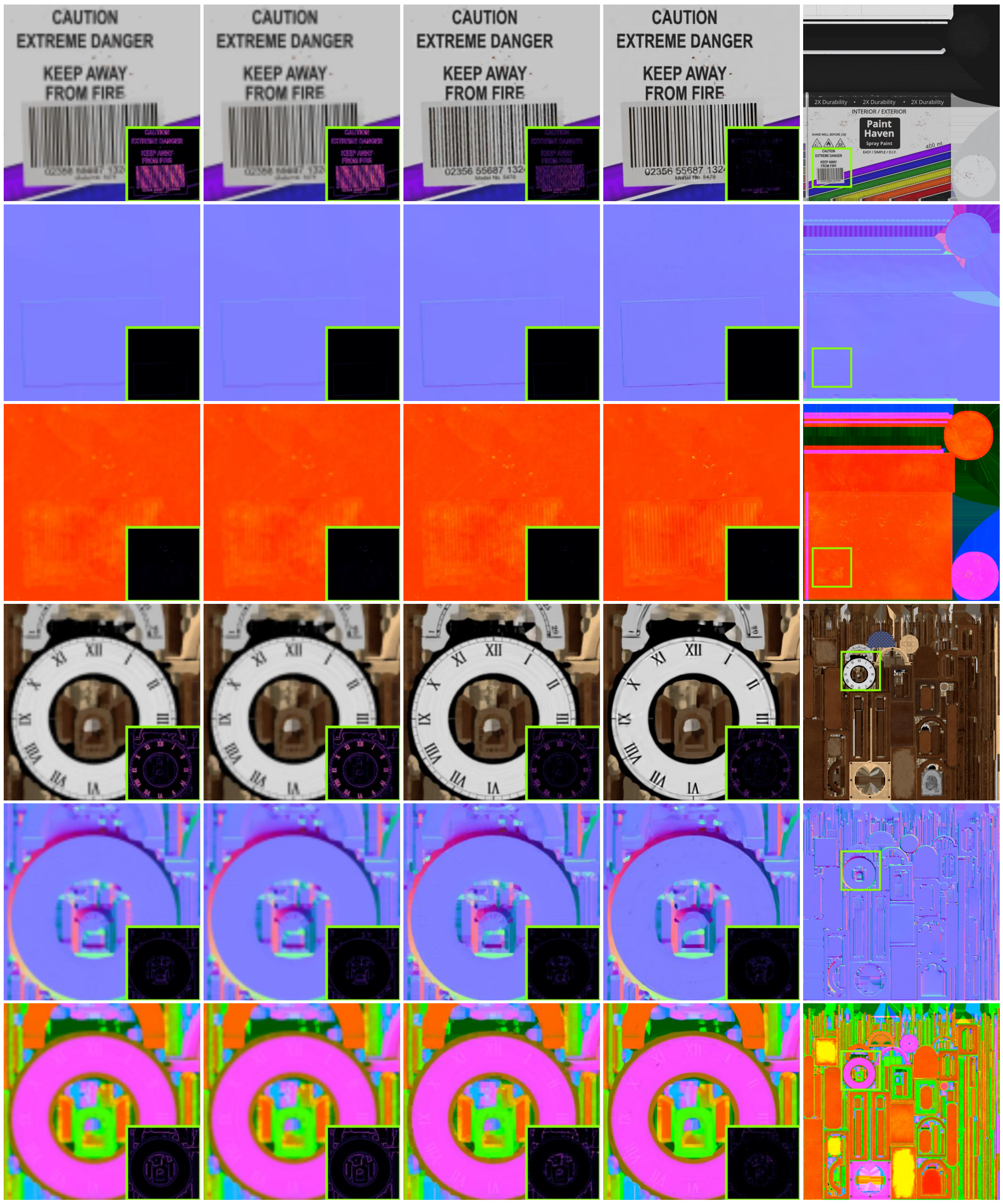


Fig. 19. *Qualitative comparison against industry-standard GPU texture compression algorithms (Section 4.5).*

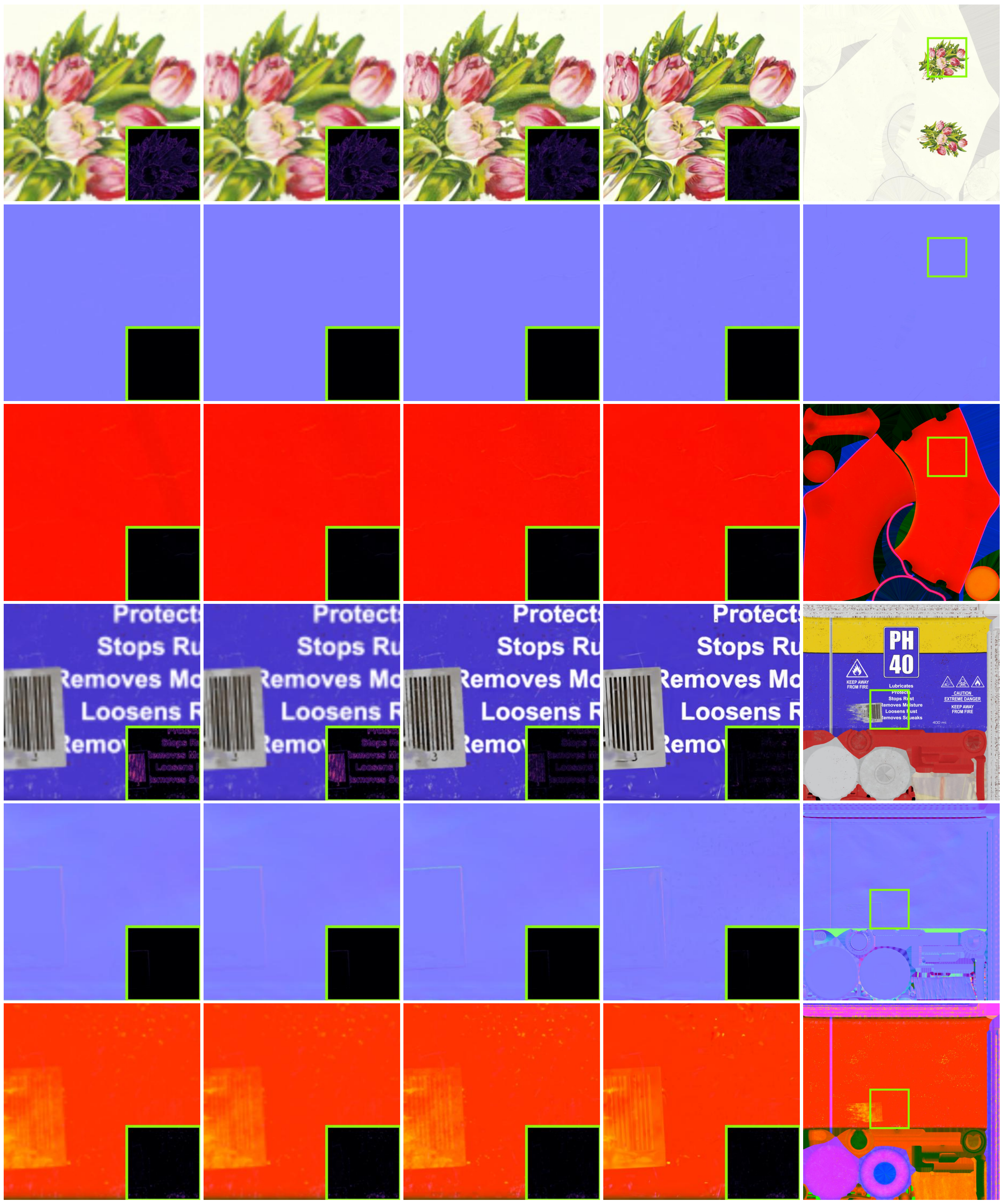


Fig. 20. *Qualitative comparison against industry-standard GPU texture compression algorithms (Section 4.5).*