

Frame Factory: Market-Aware Cloud Rendering for Large-Scale EXR Datasets

Tomáš Sikora

Reaigen s.r.o.
Slovakia

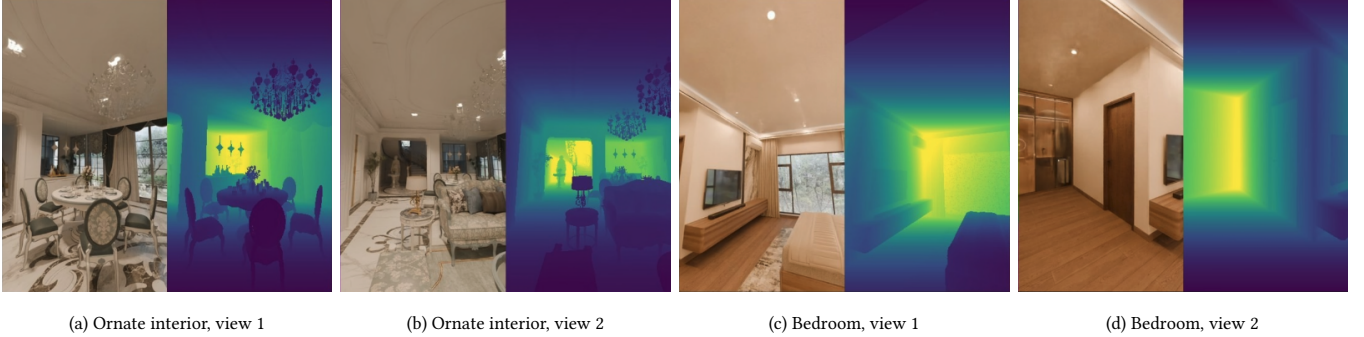


Figure 1: Paired RGB and false-color depth views from the production review interface. These original crops show interior appearance and corresponding visible-surface structure, not paired Cycles comparisons. Depth palettes may use different per-view ranges and are not a shared metric color scale.

Abstract

Producing millions of calibrated indoor frames requires affordable rendering, consistent supervision, and reliable storage delivery. We present Frame Factory, Reaigen s.r.o.’s system for large-scale EXR dataset production on rented GPU pods. Its rendering subsystem, the Reaigen glTF system, adapts NVIDIA’s Vulkan-based `vk_gltf_renderer` through a Cycles-referenced workflow with camera correspondence and G-buffer-guided validation. Frame Factory scans GPU rental offers, ranks eligible capacity by measured render-phase cost per frame, and applies a payback gate to cost-driven worker replacement. Reusable workers, camera batching, independently compressed RGB and depth, bounded staging, and verified delivery connect capture to dataset completion. Engineering records report 0.588 s and 0.940 s per color-and-depth frame on RTX 4070 Ti and RTX 3090 configurations at 16 samples with OptiX denoising. A separate production snapshot reports a 0.80 s median render-phase time for RTX 5080 jobs, 142,010 unique approved frames on storage, and \$0.123 per 1,000 produced frames under priced-job accounting. These results establish an operating production system; million-frame completion and all-in dataset cost remain beyond the reported snapshot. Renderer timing, Cycles agreement, and delivery are evaluated separately. The contribution is the integration of reference-driven rendering, market-aware resource control, and EXR delivery into one auditable production architecture.

Keywords

synthetic datasets, cloud rendering, GPU scheduling, EXR, path tracing, Vulkan, glTF, RGB-D

1 Introduction

Synthetic indoor data is valuable because rendering can expose camera poses, depth, visibility, and scene identity that are difficult to measure in a real apartment. Hypersim demonstrates the usefulness of combining authored environments with geometric supervision [13]. At production scale, however, the desired output is more than a plausible image. It is a calibrated, indexed frame record whose appearance, geometry, annotations, and provenance refer to the same approved scene.

Our practical objective is to produce millions of EXR-based frame records from authored interiors using affordable cloud GPU capacity. *Frame Factory* names the complete dataset-production and cloud-infrastructure system developed by *Reaigen s.r.o.*; the *Reaigen glTF system* names its rendering subsystem. Blender and Cycles remain the authoring and visual-reference environment. Repeated capture moves to the Reaigen glTF system, derived from NVIDIA’s Vulkan renderer [9]. glTF 2.0 provides the interchange representation while the runtime performs ray-traced image synthesis [5].

This objective has two coupled costs. The first is the cost of synthesizing a frame at an acceptable appearance and

supervision quality. The second is the cost of renting, preparing, feeding, recovering, and retiring the machines that produce it, while moving large EXR outputs to durable storage. At million-frame scale, scene-loading overhead, duplicate work, idle rental, and undrained files become production constraints. Optimizing an isolated renderer timer is therefore insufficient to optimize the finished dataset.

Frame Factory combines a specialized capture path with an empirical cloud controller. It scans available GPU rental offers, converts hourly prices into estimated cost per rendered frame using its own job measurements, and replaces expensive workers only when an explicit payback test justifies the transition. Repeated market scanning accounts for changes in the available offers after a long-lived worker is rented. The implemented policy procures on-demand RunPod capacity rather than bidding for preemptible spot instances [12].

The data path is equally important. Workers reuse renderer builds and scene packages, render multiple poses per launch, and stream EXR-based outputs through a bounded host staging area to network-attached storage (NAS). RGB and geometric data have different compression contracts. A retry remains another attempt at an existing frame identity, not a new example. Human approval is bound to the scene variant, and final completion is distinct from a successful GPU job.

This paper makes three systems contributions. First, it describes the Cycles-referenced Reaigen glTF capture subsystem, including explicit camera conversion and G-buffer-guided diagnosis. Second, it presents Frame Factory’s market-aware orchestration: empirical unit-cost ranking, payback-gated worker rotation, reusable workers, and delivery-constrained resource control. Third, it connects these mechanisms to a verified EXR production path and reports renderer, image-comparison, production, and cost observations under separate measurement boundaries. Figure 2 summarizes the combined architecture.

The evaluation documents an operating production slice and its current scale. It distinguishes implemented mechanisms and recorded results from the larger-scale validation needed for a completed million-frame corpus.

2 Related Work

Indoor data and procedural generation. Hypersim combines photorealistic, professionally authored interiors with extensive annotations [13]. BlenderProc2 provides a procedural pipeline around Blender for photorealistic data generation [3]; Kubric combines rendering and simulation in a scalable annotated-data framework [4]. These systems establish the importance of scene preparation, annotation generation, and repeatable execution. Frame Factory concentrates

on a production boundary: reusing approved authored interiors and canonical camera sequences while specializing the repeated capture backend. The focus is production infrastructure, not a new public dataset or a claim of greater downstream accuracy.

GPU rendering for synthetic data. NViSII demonstrates scriptable, GPU-accelerated photorealistic synthetic image generation [7]. NVIDIA’s Vulkan glTF renderer supplies the upstream foundation for our rendering backend [9], while Vulkan ray-tracing pipelines provide the API-level execution model [6]. OptiX supplies the denoising technology [8]. The Reaigen contribution lies in the Cycles-referenced adaptation, capture contracts, and surrounding production system. A version-matched comparison with the unmodified upstream renderer is the next step for isolating the incremental performance benefit of the Reaigen glTF modifications.

Interchange and image comparison. glTF standardizes asset representation and physically based material parameters but does not require all renderers to produce identical images [5]. Successful export is therefore the starting point for validation. We use PSNR as a direct image-error summary and FLIP as a complementary perceptual difference measure [1]. Depth correctness, temporal stability, and downstream learning utility require separate tests. Image scores alone do not establish dataset utility.

Cloud execution and data movement. RunPod exposes GPU types, pricing, availability, and pod lifecycle operations through provider APIs [14]; its billing separates compute and storage resources [15]. These infrastructure primitives do not directly estimate useful dataset output. Frame Factory connects them to its renderer measurements, reviewed capture queue, and EXR delivery constraints. The resulting policy is an empirical production heuristic for the observed workload, rather than a globally optimal scheduler or a new market mechanism.

3 Scene Transfer and Validation

3.1 A frozen scene-to-capture contract

The input to capture is a reviewed scene package, calibrated intrinsics, and one or more enumerated camera sets. A camera set may represent a walk through the navigable interior or a separate coverage trajectory. Frame Factory’s preparation workflow preserves authored scene information in the Blender source and produces the runtime package outside the cloud rendering loop. Production workers do not need to execute Blender to render every frame [12].

Source preservation and runtime equivalence are separate requirements. The source scene retains its authored materials and textures, while the runtime conversion is evaluated against its supported representation and reference output. Arbitrary Blender node graphs do not necessarily

translate exactly to glTF materials. Unsupported shading behavior, emissive interpretation, normal smoothing, and light-intensity conventions can alter appearance even when exported meshes occupy the correct positions.

For a frozen package, a capture is identified by the scene, package version, lighting variant, camera set, and pose index. The production database binds review evidence to package and lighting identity. Re-rendering the same pose is a new attempt at the same required capture, not a new unique dataset example. Output accounting separates these attempts from unique dataset growth.

There are two distinct operating modes. A *reference-faithful* mode aims to reproduce the approved source scene. An *enhanced variant* may intentionally change lights, materials, or geometry. The implementation supports such modifications in a copied package; some options append geometry for lighting fixtures or architectural additions. Original binary-payload preservation is consequently distinct from an unchanged final scene. Each enhanced variant needs its own identity and review, plus a matching Cycles reference when included in a parity comparison [12].

3.2 Camera conventions and correspondence

A valid comparison begins with the same camera. The source pipeline must reconcile Blender coordinates, the dataset coordinate system, and COLMAP-compatible camera files. COLMAP represents the world-to-camera transform as

$$\mathbf{x}_c = \mathbf{R}\mathbf{x}_w + \mathbf{t}, \quad \mathbf{C}_w = -\mathbf{R}^\top \mathbf{t}, \quad (1)$$

with camera axes pointing right, down, and forward [2]. The documented Blender-to-dataset axis mapping is

$$\mathbf{A}(x, y, z)^\top = (x, z, -y)^\top. \quad (2)$$

The inverse mapping and the camera up/down sign must be applied consistently when constructing the renderer’s position and orientation vectors. Numerical camera residuals evaluate this conversion; scene geometry is assessed separately.

Frame correspondence also requires explicit indexing. The evaluation code maps factory previews to runtime poses using the trajectory’s motion-frame metadata. A one-based preview filename is not treated as a zero-based pose index. This prevents an adjacent-frame comparison from being mistaken for a renderer difference [12].

Depth requires an equally explicit convention. For a homogeneous pixel coordinate $\tilde{\mathbf{u}}$ and intrinsics $\mathbf{K}$, let $\mathbf{q} = \mathbf{K}^{-1}\tilde{\mathbf{u}}$, normalized so that $q_z = 1$. Camera-axis depth z and Euclidean ray range r back-project differently:

$$\mathbf{x}_c = z\mathbf{q} \quad \text{or} \quad \mathbf{x}_c = r \frac{\mathbf{q}}{\|\mathbf{q}\|_2}. \quad (3)$$

A floating-point channel named Z does not resolve this distinction. Depth definition, unit scale, invalid-pixel policy, and treatment of transparent surfaces belong in the data schema. These semantics are not established by the retained false-color review views and remain explicit requirements for geometric validation.

3.3 G-buffer-guided development

The development strategy uses intermediate buffers to localize mismatches before adjusting final RGB appearance. We formalize the diagnostic state at pixel u as

$$G(u) = \{v, z, \mathbf{p}, \mathbf{n}_g, \mathbf{n}_s, \mathbf{a}, m, \alpha\}, \quad (4)$$

where v is validity, z is depth, $\mathbf{p}$ is position, $\mathbf{n}_g$ and $\mathbf{n}_s$ are geometric and shading normals, $\mathbf{a}$ is a linear-space material color, m is surface identity, and α denotes coverage. Equation 4 defines the validation decomposition; it does not imply that every channel is exported in every production file.

Diagnosis follows a fixed order. Camera and validity disagreements are resolved first. Position, depth, and surface identity then test scene correspondence. Normal and material buffers localize shading-transfer differences. Once these agree, the final RGB residual helps distinguish lighting, light transport, sampling, denoising, and display-transform differences. Normal comparisons require a consistent coordinate system and must distinguish geometric normals from normal-mapped shading normals.

For jointly valid pixels Ω and unit-length normals, representative depth and normal diagnostics are

$$E_z = \text{median}_{u \in \Omega} \frac{|z_V(u) - z_C(u)|}{\max(\epsilon, |z_C(u)|)}, \quad (5)$$

$$E_n = \text{median}_{u \in \Omega} \arccos[\text{clamp}(\mathbf{n}_V(u) \cdot \mathbf{n}_C(u), -1, 1)]. \quad (6)$$

The subscripts indicate Vulkan and Cycles reference outputs. Absolute errors, percentile tails, invalid-mask disagreements, and edge-specific residuals should accompany these summaries. The retained evaluation reports camera checks and final-image comparisons. Matched raw G-buffers and numerical values for Equations 5 and 6 are not included, so geometry parity remains a defined validation task rather than a reported numerical result.

3.4 Appearance and reference review

The rendering lane records the display look and carries the Blender OpenColorIO configuration into the worker tools. Matching the intended AgX transform and scene exposure is necessary for comparing display-referred PNGs [12]. Material transfer must additionally respect which glTF textures represent sRGB color and which encode linear quantities [5].

Normals deserve particular attention in imported architectural content. The engineering records describe ceiling

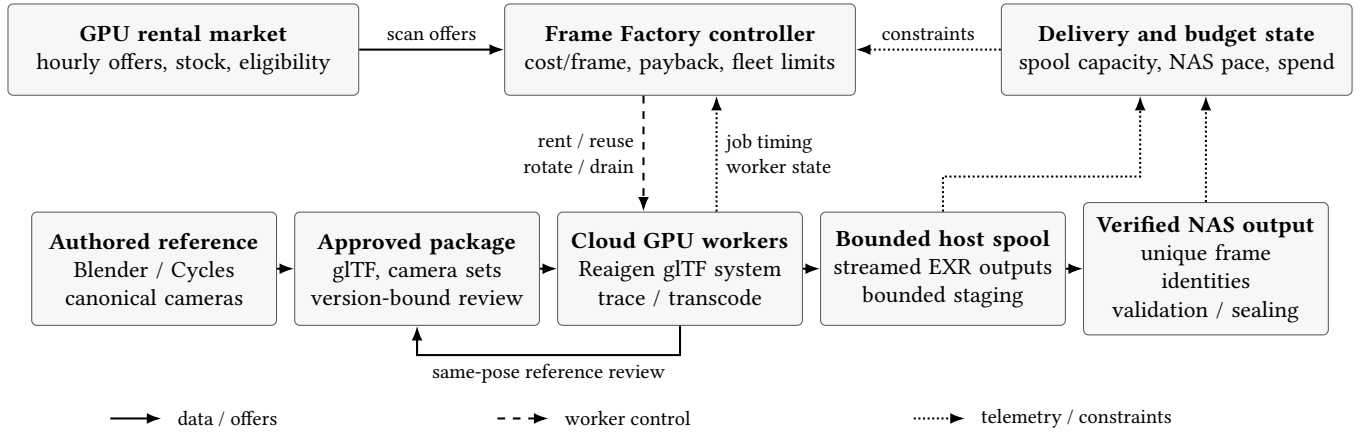


Figure 2: Frame Factory separates the data path from a feedback-driven cloud controller. The Reaigen glTF system operates inside reusable GPU workers. Market offers and measured job pace drive unit-cost ranking; budget and delivery state constrain the fleet. RGB/depth transcoding begins on workers before transfer through the host spool. Reference review, landed-frame verification, and dataset sealing are distinct checks. The renderer fork and control plane are maintained as separate components.

artifacts associated with broadly smoothed faces and an optional architectural conversion mode. Because this mode can also alter material and appearance settings, it is evaluated as a scene variant rather than as an isolated normal-repair operation. Reference comparisons must use the exact resulting variant to attribute observed appearance changes correctly.

Review views are deliberately chosen to reveal the scene. Candidate selection combines camera positions, sparse-point visibility, and spatial spread; rendered depth is then used to avoid previews dominated by an immediately adjacent wall. Figure 3 shows the resulting multi-view presentation. This supports human approval, while systematic benchmarking requires a separate sample that includes close surfaces, reflective objects, thin structures, windows, and difficult trajectory segments. Curated review views should not be treated as a random sample of capture conditions.

4 The Reaigen glTF Capture System

4.1 A headless rendering backend

The configured production profile uses portrait images at 1080×1920 pixels, 16 samples per pixel, and OptiX denoising. The small-scene timing report explicitly records the sample count and denoiser. Reproducing each reported run additionally requires its resolution and renderer flags; lane defaults alone do not establish the configuration of every retained measurement [12].

The Reaigen glTF system adapts NVIDIA’s renderer and uses Vulkan for ray-tracing execution [6, 9]. Scene conversion, pinned renderer metadata, and shader patches are supplied by a separate rendering repository. The control plane

computes a build key from the relevant build configuration, pinned metadata, and patch files, then stages renderer tools and color-management configuration for workers [12]. The control-plane snapshot documents this integration mechanism. The separately maintained renderer patches and exact revisions for each reported run are additional requirements for renderer-level reproduction; the present evidence establishes the integrated workflow rather than a new sampling or denoising algorithm.

The specialization separates authoring from repeated capture. Runtime packages and canonical cameras are prepared in advance. A worker loads a scene, renders a range of poses, records renderer timing and camera evidence, and emits image files without invoking the authoring application for each pose. Denoising is part of the measured capture configuration. Accordingly, the 16-sample results characterize the denoised output path, not undenoised convergence at that sample count.

4.2 Amortizing initialization

A long-lived worker reuses its renderer build and caches scene packages, while each renderer launch still incurs scene and shader initialization. The worker code groups 250 poses per launch within larger jobs, commonly 1,000 poses. Reuse therefore operates at both worker and launch boundaries; it does not require a scene to remain GPU-resident across the entire corpus.



Figure 3: Six views of one interior from the scene-review interface. Different positions and orientations support production approval. These original qualitative crops show Reaigen glTF runtime output; paired Cycles fidelity and temporal consistency are separate evaluation tasks.

For N frames and a launch capacity B , a simplified sequential compute-side model is

$$t_{\text{amortized}}(N, B) = t_f + \frac{\lceil N/B \rceil t_{\text{load}} + t_{\text{prepare}}}{N}, \quad (7)$$

where t_f includes the per-frame capture work, t_{load} is per-launch initialization, and t_{prepare} is one-time preparation within the chosen timing boundary. This model explains why a fast per-frame counter can coexist with a slower ten-frame cold-start job. It is an explanatory model rather than a fit to the reported workload.

The engineering notes record approximately 26% loading overhead for 100-pose launches versus approximately 12% for 250-pose launches in an observed 1,000-frame workload. These operational observations motivate larger launch batches; a controlled multi-scene ablation remains necessary to generalize the effect. The same records describe cached renderer deployment replacing a roughly two-minute compilation step with a short unpack step. Build reuse and camera

batching address different initialization costs and should be evaluated separately [12].

4.3 Different guarantees for appearance and depth

Each expected capture has an RGB PNG, a pass EXR, and an RGB–depth EXR. The transfer path reduces output volume while retaining distinct accuracy policies. In the implemented transcoder, the RGB–depth file is rewritten as a multipart EXR. Its RGB part uses half-precision channels and DWAB compression with DWA compression level 45. Its depth part uses 32-bit floating-point Z, choosing the smaller of PIZ and ZIP encodings for that frame [12].

The compression contract is lossy for RGB and lossless for the selected depth encodings [10]. The implementation reads candidate depth back and compares the bit patterns of decoded float32 samples with the input array. A candidate is accepted only when

$$\text{bits}(Z_{\text{decoded}}) = \text{bits}(Z_{\text{input}}) \quad \text{for every depth sample.} \quad (8)$$

On failure, the original output is retained. This verifies depth preservation through transcoding; the physical correctness of input depth is a separate geometric-validation requirement.

The pass EXR is also recompressed and checked against its decoded input. Lossless recompression preserves that input but cannot restore precision lost in an upstream representation. In particular, OpenEXR’s PXR24 representation can reduce precision of floating-point channels [10]. The PNG is retained independently. Image agreement measured on PNGs therefore describes a different representation from the lossy EXR RGB that a downstream consumer may use, and the two should be assessed separately.

5 Market-Aware Cloud Infrastructure

5.1 Ranking available capacity by frame cost

A pod’s hourly rental price is not its unit production cost. Let p_g be an eligible offer’s price in dollars per hour and $\hat{t}_g$ the measured render-phase seconds per frame for its GPU type. Frame Factory estimates

$$\hat{c}_g = \frac{p_g \hat{t}_g}{3600}, \quad \hat{C}_{g,1000} = 1000 \hat{c}_g = \frac{p_g \hat{t}_g}{3.6}. \quad (9)$$

The first quantity is dollars per rendered frame; the second is dollars per 1,000 frames. This conversion allows the controller to compare useful render output rather than hourly price alone, which can favor a slower card with a higher cost per frame.

The implementation obtains $\hat{t}_g$ from per-GPU medians of job render-stage duration divided by frame count, over up

to 300 recent completed jobs. Retained engineering measurements supply initial values. Unmeasured types inherit the slowest available estimate for routine placement, providing a conservative default. The scheduler refreshes the market every 300 s and checks GPU eligibility, memory, price limits, availability, and placement policy [12]. Availability is an input to placement; a subsequent rental request may still fail [14].

This estimate measures render-phase economics rather than all-in dataset cost. Startup, idle, transfer, storage, failed-attempt, and approval costs are handled through worker reuse, rotation gates, delivery constraints, and separate rental accounting, rather than being fully represented in Equation 9. Per-type medians reduce sensitivity to individual difficult jobs, while scene-mix bias and host-to-host variation remain relevant when interpreting GPU rankings.

5.2 Payback-gated worker rotation

A long-lived worker may cease to be economical when a better offer becomes available. Frame Factory compares an eligible incumbent 0 with a different, previously measured GPU type 1 using the same per-type timing estimates. The candidate must satisfy

$$\widehat{c}_1 \leq (1 - m)\widehat{c}_0, \quad m = 0.20. \quad (10)$$

The recorded policy also requires a worker age of at least 30 minutes, a five-minute rotation cooldown, and at least two hours of queued work estimated at the current fleet’s aggregate render pace [12].

To model transition cost, the policy uses a 300 s allowance t_{boot} for preparation and re-staging. Its simplified rental payback calculation is

$$C_{\text{boot}} = \frac{p_1 t_{\text{boot}}}{3600}, \quad (11)$$

$$\Delta_{\$} = (\widehat{c}_0 - \widehat{c}_1) \frac{3600}{\widehat{t}_1}, \quad (12)$$

$$H_{\text{payback}} = \frac{C_{\text{boot}}}{\Delta_{\$}}. \quad (13)$$

Here C_{boot} is dollars, $\Delta_{\$}$ is estimated dollars saved per hour when producing at the candidate’s pace, and H_{payback} is hours. Rotation requires payback within two hours. The decision compares equal-output costs rather than the difference between hourly prices. The fixed preparation allowance is a policy assumption, not a measured upper bound on every startup.

Replacement is make-before-break: the new worker must become ready before the incumbent drains its current chunk and terminates. This can temporarily add one worker above the steady-state ceiling, while avoiding interruption of a render solely to obtain a lower offer. Only one replacement proceeds at a time. Failed replacements back off for 10, 20, and 40

minutes, then at most an hour. A separate, rate-limited probe path measures selected untried GPU types; its optimistic eligibility estimate is used for exploration, not reported as a measured unit cost [12].

The controller records the last market snapshot and decision reason. An operator can therefore distinguish price-driven rotation from capacity growth, failure replacement, or a storage-induced drain. These records make the reactive policy inspectable without a future-price forecast. Controlled policy comparisons remain a next evaluation step.

5.3 Reusable workers and fault containment

Workers rent and initialize a pod once, then claim successive jobs through database row locks. Claiming prefers jobs with fewer previous attempts and cached scene packages, while avoiding scenes that already failed on that worker. Keeping recently used packages on pod storage reduces repeated uploads; renderer build reuse and per-launch pose batching address separate initialization costs [12].

Frame Factory monitors progress in the output files rather than treating an active process or a changing log as sufficient evidence of productive rendering. Stalls retain per-attempt diagnostics. Failed jobs return behind fresh work or enter an investigation state; repeated failures on multiple workers can hold an affected scene without blocking unrelated scenes. Worker heartbeats and orphan cleanup recover from a lost scheduler or abandoned pod. A reused worker eventually drains on lifetime, idle, operator, or resource-control conditions rather than remaining rented indefinitely.

Execution is recoverable, with capture identity maintained independently of task attempts. Queue deduplication checks already completed pose ranges as well as request keys. This supports accurate output accounting even when a task runs more than once. Failure statistics used by the controller are operational signals; a selected recent-success window is not an unconditional reliability estimate for the entire workload.

5.4 EXR streaming and delivery-constrained scaling

The output path is a distributed pipeline. RGB/depth transcoding runs on the worker alongside rendering. Ready files move through a bounded host spool and then to the NAS. A worker may proceed while earlier chunks finish their storage passage, so NAS duration cannot be added to render duration as though all stages were serial [12].

For P workers with effective per-frame production times t_i , mean delivered bytes per frame d , and sustainable storage-path bandwidth B_{net} , an upper bound is

$$R_{\text{stored}} \leq \min \left\{ \sum_{i=1}^P \frac{1}{t_i}, \frac{B_{\text{net}}}{d}, R_{\text{transcode}}, R_{\text{verify}} \right\}. \quad (14)$$

All terms are frames per second. Shared resources, bursty launches, and idle periods can lower the realized rate further. Additional low-cost GPUs increase completed output only while other stages retain sufficient capacity.

The fleet planner combines configured pod limits, remaining budgets, demand, measured worker output, recent outcomes, and storage safeguards. The base link estimate uses bytes moved over render plus drain time, rather than nominal GPU throughput. Delivery and disk-reservation safeguards can further constrain launches. Pausing new work and draining workers are distinct from interrupting an in-flight render. Table 4 shows a recorded planner state in which delivery and the configured pod ceiling, rather than the base link allowance or demand, bind the fleet.

These mechanisms make compression a scheduling decision as well as a file-format choice. Reducing bytes per frame without altering decoded depth can admit more useful rendering work under the same delivery capacity. Lossy RGB and multipart-reader compatibility remain explicit dataset contracts: a smaller EXR must still satisfy the intended consumer’s representation and quality requirements.

5.5 Validation is not sealing

The delivery validator checks required frame files, camera coverage, and associated metadata. Sealing requires the expected captures to be present, blocking job states to be resolved, and validation to pass. Frame-file hashes are recorded in a manifest and the written manifest is verified [12]. A landed frame and a sealed dataset are consequently distinct production milestones.

The current manifest records frame-file integrity; it does not establish a cryptographic commitment to every auxiliary file. Complete reproducibility also requires renderer, conversion, color-management, and metadata identities. At the reported snapshot, frame delivery was active and substantial unique output had reached storage, while dataset-level sealing had not yet been completed. Section 6 reports both stages explicitly.

6 Evaluation

6.1 Evidence and measurement boundaries

The results come from two existing records: a 19 September 2026 engineering evaluation retained in the 20 September source archive, and production telemetry and scene-review captures dated 21 September [11, 12]. The evaluation reports

Table 1: Recorded rendering times. The Vulkan entries are color-and-depth per-frame measurements from a ten-frame test. The Cycles entry is a factory record for the same scene, with different or undocumented hardware and settings.

Recorded configuration	Seconds/frame
Reaigen glTF, local RTX 4070 Ti	0.588
Reaigen glTF, cloud RTX 3090	0.940
Factory Cycles, GPU not recorded here	10.5–12.3

Vulkan test: 16 spp, OptiX-denoised, ten walk poses. Cycles sampling, denoising, hardware, and full timing scope are not established by the retained summary; this is not a matched-quality speedup benchmark.

recorded measurements rather than a newly rerun benchmark suite. The retained archive does not contain raw reference EXRs, the external renderer patches, or per-frame logs sufficient to independently recompute all aggregate values. The observation dates are treated separately, without assuming identical renderer builds.

Three timing boundaries remain distinct. *Renderer-reported frame time* is the internal color-and-depth figure in the small-scene report. *Render-phase time per frame* in the dashboard is job render-stage wall time divided by its frame count; it includes launch-side work within that stage and is not isolated GPU kernel time. *Stored throughput* counts frames that reached the NAS over a wall-clock interval. Provisioning, package transfer, overlapping transcoding, idle periods, and final delivery affect these quantities differently.

Likewise, cloud/local agreement measures the same rendering pipeline on different machines, while Cycles agreement compares different renderers at corresponding poses. The former tests deployment consistency; the latter tests reference fidelity. Strong deployment agreement is not a renderer-parity measurement.

6.2 Recorded frame times

Table 1 summarizes the engineering test: ten walk frames from one scene at 16 samples per pixel with OptiX denoising and EXR output. The recorded color-and-depth time is 0.588 s per frame on a local RTX 4070 Ti and 0.940 s on a cloud RTX 3090. The same scene’s factory Cycles records give 10.5–12.3 s per frame [12].

Dividing the recorded Cycles interval by the Vulkan values gives nominal ratios of 11.2–13.1 for the cloud configuration and 17.9–20.9 for the local configuration. These ratios describe the recorded times, not controlled speedups at matched hardware and image quality. The faster local observation across different GPU classes also illustrates why this comparison is not a hardware ranking.

Table 2: Two image-agreement experiments from the engineering report. Cloud/local agreement measures deployment consistency; the Cycles comparison measures reference fidelity. PSNR is higher-is-better and mean FLIP is lower-is-better.

Comparison	PSNR (dB)	Mean FLIP
Cloud vs. local, same pipeline	55.5–56.7	≤ 0.01
Vulkan vs. Cycles, three poses	26.1–32.7	0.09–0.14

Values are reported ranges, not confidence intervals. The comparator operates on display-referred RGB PNGs.

Initialization changes the picture for short jobs. The report records a 65 s render phase for the ten-frame cold-start cloud run, including about 50 s of scene loading and shader compilation. A subsecond per-frame renderer counter therefore coexists with approximately 6.5 s per frame when that render stage is divided by only ten outputs. Provisioning and transfer add further latency. These observations motivate worker reuse and pose batching while preserving the distinction between frame time and job latency.

6.3 Image and camera agreement

Table 2 separates consistency from reference fidelity. The cloud/local comparison gives 55.5–56.7 dB PSNR and mean FLIP at most 0.01. The three available Cycles preview comparisons give 26.1–32.7 dB PSNR and mean FLIP 0.09–0.14 [12]. The latter quantifies the remaining reference differences. The evaluation does not establish pixel identity or a universal perceptual-equivalence threshold for those results.

The comparator decodes PNG RGB into $[0, 1]$, computes mean squared error over the image, and reports

$$\text{PSNR} = 10 \log_{10} \frac{1}{\text{MSE}}. \quad (15)$$

Its FLIP call uses the LDR mode of the evaluator [1, 12]. These metrics describe display-referred images, not linear HDR radiance. The current comparator can resize the test image to the reference dimensions when they differ. A controlled benchmark should reject a resolution mismatch or record resampling as an explicit experimental condition; this behavior is a comparability consideration for extending the evaluation.

The engineering report also records a worst camera discrepancy of 0.0002 mm in position and $0.67 \mu\text{rad}$ in orientation. These are conversion/check residuals, not scene-geometry accuracy estimates. Validated ray-depth semantics and exact first-hit correspondence require matched raw buffer comparisons and a declared transparent-surface policy.

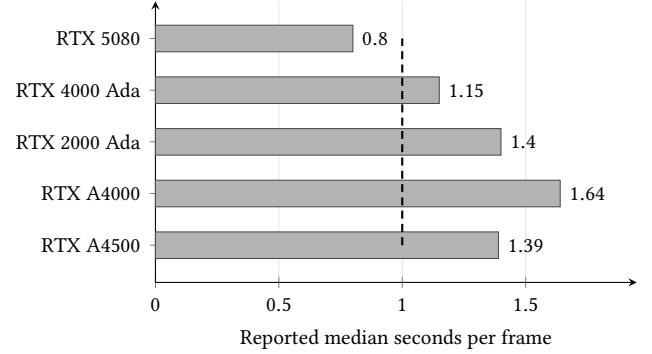


Figure 4: Production-dashboard medians of job render-phase seconds divided by frames. The dashed line marks the 1 s design target. These operational summaries retain the original reported values; job counts, confidence intervals, and scene-balanced hardware comparisons are not available in the snapshot.

The review imagery in Figures 1 and 3 illustrates interior output and multi-view review coverage. Full-resolution paired Cycles/Vulkan outputs are not included in the retained review imagery, so no new error map or additional image metric is inferred from those figures.

6.4 Production render-phase observations

Figure 4 reproduces the five GPU-specific medians in the production timing chart. An observation in that chart is one job’s render-phase duration divided by its number of frames. The dashboard-reported median for RTX 5080 jobs is 0.80 s per frame. The corresponding values are 1.15 s for RTX 4000 Ada, 1.40 s for RTX 2000 Ada, 1.64 s for RTX A4000, and 1.39 s for RTX A4500 [11].

The RTX 5080 result establishes a subsecond median render phase for the observed production slice. The other medians show variation across the rented pool. Jobs differ in scene, launch overhead, worker conditions, and potentially run configuration. Without raw paired workloads and job counts, the summaries characterize operating conditions rather than controlled comparative GPU performance.

The same production view shows a recent-ten-chunk pace of 1.29 s per frame and a last-six-hour output rate of 4,465 frames per hour. These have different aggregation windows and denominators. They describe recent render pace and fleet-level output, respectively, rather than one shared throughput statistic.

6.5 Stored output and cost accounting

Table 3 reports the captured production state. Of 384,652 required frames across 51 approved scenes, 142,010 unique approved frames had reached the NAS, corresponding to 36.9%.

Table 3: Production snapshot, 21 September 2026. Recorded operational progress and frame-cost statistics, with dataset completion and accounting scope identified separately.

Quantity and scope	Recorded value
Approved scenes in this snapshot	51
Required frames in approved scenes	384,652
Unique approved frames on NAS	142,010
Rendered frames, including tests	166,876
Landed in the previous 24 hours	63,614
Landed in the previous hour	4,236
Sealed datasets	0
Priced-job cost / 1,000 produced frames	\$0.123
Worker rental / 1,000 worker outputs	\$0.130

Cost values use different accounting scopes. Neither is a measured all-in cost per 1,000 unique, accepted, sealed dataset frames.

The separate count of 166,876 rendered frames includes tests and is not the unique approved-delivery count [11].

The priced-job statistic is \$0.123 per 1,000 produced frames over 77 priced jobs in 24 hours and excludes unallocated rental overhead. The worker-rental statistic is \$0.130 per 1,000 produced worker frames, including estimated startup, idle, and failed-worker rental. It is an operational estimate rather than reconciled provider billing or accepted-delivery cost. Conversion, preparation, storage, administration, and duplicated attempts require separate accounting before a complete dataset unit cost can be calculated.

The interface records a median recent chunk-passage rate of 20.9 MB/s into the NAS, while the last-day average is 5.1 MB/s against a configured 80 MB/s link limit. The former describes active chunk transfers; the latter covers a longer wall-clock interval. Neither establishes physical link saturation. Worker limits, delivery safeguards, burstiness, and idle periods can lower sustained output even when the configured link limit is higher.

The snapshot records zero sealed datasets. It therefore demonstrates substantial delivered frame volume and operating orchestration, with final dataset-level validation and sealing still outstanding. Approval of the 51 scenes is a preparation milestone, not evidence that 51 complete datasets had passed final integrity checks or become a released corpus.

6.6 Observed resource decisions

The displayed constraints illustrate why marketplace scanning and data delivery must be considered together. A budget allowance of 56 pods does not imply that renting 56 pods would increase useful output when the active delivery safeguard and configured ceiling both allow four [11]. Likewise, a

Table 4: Fleet limits in the production captures. Both binding limits admit four pods. Values are controller allowances rather than measured hardware capacities or results of a concurrency-scaling experiment.

Displayed constraint	Pods	Status
NAS link allowance	19	Non-binding
Budget allowance	56	Non-binding
Demand allowance (jobs waiting)	241	Non-binding
Configured pod ceiling	4	Binding
NAS delivery allowance	4	Binding

Displayed recent worker output: 4.0 MB/s. Per-pod price cap: \$0.50/h. The records are dashboard snapshots, not synchronized raw event logs.

base link allowance of 19 does not indicate physical-link saturation by the smaller fleet. Table 4 records these controller allowances.

The source records the unit-cost and rotation policies, and the interface exposes their operating state. A controlled counterfactual against a static GPU choice or an hourly-price-only scheduler is not included in the retained evaluation. The \$0.123 priced-job statistic is therefore an observed cost level, not a measured saving attributable specifically to market scanning. The \$0.130 worker-rental statistic includes a broader overhead scope; both remain distinct from accepted-and-sealed dataset unit cost.

6.7 Planning for millions of EXR frame records

The recorded capacity plan explicitly considers five million frames [12]. A frame record is the complete output set for one camera pose, including its EXRs and preview; individual files are not counted as additional frames. Taking the plan’s approximately 7–8 MB per frame as decimal units gives 35–40 TB of final frame files. At a sustained 80 MB/s, the transfer-only lower bound is approximately 5.1–5.8 days. At 1.1 s per frame, aggregate rendering work would be approximately 1,528 GPU-hours before additional lifecycle costs.

For a homogeneous illustrative fleet, this distinction can be written as

$$T_{\text{plan}} \geq \max\left(\frac{Nt_f}{P}, \frac{Nd}{B_{\text{net}}}\right), \quad (16)$$

where N is the required number of frames and t_f is the assumed compute-side frame time. This is capacity planning rather than a completed five-million-frame benchmark. It excludes source packages, intermediate files, reports, retries, replicas, startup, and idle intervals. The bandwidth assumption is sustained delivery over the production run, not a brief peak transfer rate.

This planning scale motivates the architecture: amortize fixed costs on warm workers, select GPU offers by frame economics, preserve decoded depth during output reduction, and keep temporary staging bounded. The measured milestone remains 142,010 unique approved landed frames; the five-million-frame scenario defines planning requirements.

7 Discussion and Limitations

7.1 What “Cycles-referenced” establishes

The engineering result is a specialized runtime that moves approved interior capture into the subsecond regime in measured configurations, supported by explicit reference comparison and geometric-supervision outputs. Its value lies in the integrated production workflow: stable camera identities, interpretable diagnostics, recoverable execution, and verified delivery alongside efficient rendering. A new light-transport algorithm is not required for this systems contribution.

G-buffer agreement is a useful geometric diagnostic but does not establish radiometric equivalence. Light sampling, visibility along secondary rays, BSDF approximations, emission conventions, transmission, denoising, and the display transform can produce different RGB images even when first-surface geometry agrees. Review-scale appearance also does not quantify temporal instability, biased indirect illumination, softened details, or material-dependent differences. These properties require a broader paired evaluation than the three retained Cycles comparisons.

Data-production requirements also differ from cinematic reference rendering. Denoised low-sample appearance may be acceptable for one learning task and harmful for another. Lossy EXR RGB may preserve a useful training signal while altering fine texture. Downstream acceptability and model improvement have not been measured in the present records. The chosen configuration is therefore an operating point with documented representation and timing boundaries, rather than a universal dataset-quality guarantee.

7.2 A controlled extension of the evaluation

The next renderer-level validation is a three-way comparison: Cycles, the pinned unmodified NVIDIA baseline, and the Reaigen gITF fork. Each must use the same GPU, image resolution, source scene, camera poses, and declared scene variant. Material and lighting compatibility should be checked before timing. Sampling, bounce limits, denoising, exposure, output channels, and file-writing scope must be recorded explicitly. Separate runs should measure steady-state frame time and end-to-end job latency.

Equal samples per pixel is not equal image quality. The extended study should report time-to-error curves against a declared high-quality reference, using both linear-space errors and display-referred FLIP. Aggregation by scene as

well as by frame prevents a large easy sequence from dominating a smaller difficult one. Repeated runs and dispersion estimates are needed to evaluate hardware ranking, scene sensitivity, and performance reliability beyond the current operational summaries.

The geometric study should include matched depth and position buffers, geometric and shading normals, surface correspondence, and invalid-pixel masks. Tests should declare depth semantics and include silhouettes, transparent surfaces, normal-mapped regions, and grazing views. Sequence-level tests should examine camera-conditioned reprojection and appearance stability over time. A downstream reconstruction or depth-learning experiment would then test whether the faster capture path preserves task utility, complementing the image-level measurements.

7.3 Reproducibility and scope

The control-plane snapshot is identified in Reference [12], and the production and review records in Reference [11]. The figures retain the original review crops and operational chart. Full renderer-level reproduction additionally requires the external renderer’s pinned revision and patches, scene-conversion settings, color-management assets, raw timing records, and paired Cycles/runtime files. The current evidence is an internal engineering and production record, not an independently replicated benchmark release.

The method targets static authored interiors. Animated geometry, physically validated sensor noise, motion blur, rolling shutter, and improved generalization to real phone captures are outside the demonstrated scope. Human-approved previews establish a scene-review decision, not physical correctness of every imported asset or usability of every trajectory frame. Publication and redistribution of source interiors and figures require appropriate asset permissions; this paper does not establish a public license for those assets.

7.4 Evaluating the infrastructure contribution

A controlled scheduler comparison should replay the same approved frame workload under a fixed GPU policy, a lowest-hourly-price policy, and Frame Factory’s unit-cost policy, with and without rotation. The offer history, placement failures, configuration, startup times, byte volumes, and delivered frame identities must be retained. Relevant outcomes are total rental, unique accepted output, completion time, and idle or failed expenditure. Independent live runs encounter different markets; offline replay must therefore be grounded in measured service times and availability.

The current estimator pools recent jobs by GPU type rather than conditioning on scene complexity, cache state,

or host transfer behavior. Its fixed startup allowance and conservative unknown-card estimates prioritize transparent decisions. Delivery-aware cost estimation and controlled policy ablations are the next steps for quantifying incremental savings. Likewise, multi-million-frame completion must be established through unique validated output and final dataset sealing, rather than by extrapolating a short-run rate.

8 Conclusion

Frame Factory is Reaigen s.r.o.’s integrated cloud infrastructure for large-scale spatial dataset production. Its Reaigen glTF system moves Cycles-referenced repeated capture to an adapted NVIDIA Vulkan renderer. The surrounding controller scans GPU rental offers, ranks them by empirical frame cost, reuses workers, and gates replacement on predicted payback. Bounded staging, bit-preserving depth transcoding, retry identity, and verified delivery connect rendering to durable output.

Recorded configurations achieve 0.588–0.940 s per color-and-depth frame. The separate production snapshot reports a 0.80 s median render-phase time for RTX 5080 jobs, 142,010 unique approved landed frames, and \$0.123 per 1,000 produced frames under priced-job accounting. These results establish the functioning combined system and its observed operating economics. Cycles fidelity, total rental accounting, and final dataset completion retain their distinct measurement boundaries.

Scaling EXR production requires rendering and infrastructure to be designed together. Frame Factory treats the useful unit as an indexed frame delivered under explicit cost, fidelity, and completion contracts, rather than an isolated render or a rented GPU-hour. Matched-quality renderer experiments, controlled scheduler evaluation, and sealed larger-scale production runs are the next milestones for quantifying that architecture’s performance and dataset utility.

Acknowledgments

The rendering backend builds on NVIDIA’s open-source Vulkan renderer and existing denoising technology. The system also depends on Blender, glTF, OpenEXR, and their surrounding open-source ecosystems. We acknowledge these foundations alongside Reaigen s.r.o.’s capture, orchestration, and delivery engineering. These acknowledgments do not imply endorsement by the upstream projects.

References

- [1] Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. 2020. FLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3, 2, Article 15 (2020), 23 pages. <https://doi.org/10.1145/3406183>
- [2] COLMAP contributors. 2026. Output Format. COLMAP documentation. <https://colmap.github.io/format.html> Accessed 22 September 2026.
- [3] Maximilian Denninger, Dominik Winkelbauer, Martin Sundermeyer, Wout Boerdijk, Markus Knauer, Klaus H. Strobl, Matthias Humt, and Rudolph Triebel. 2023. BlenderProc2: A Procedural Pipeline for Photorealistic Rendering. *Journal of Open Source Software* 8, 82 (2023), 4901. <https://doi.org/10.21105/joss.04901>
- [4] Klaus Greff, François Belletti, Lucas Beyer, Carl Doersch, et al. 2022. Kubric: A Scalable Dataset Generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. <https://doi.org/10.1109/CVPR52688.2022.00373>
- [5] Khronos Group. 2017. glTF 2.0 Specification. Technical specification. <https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html> Accessed 22 September 2026.
- [6] Khronos Group. 2026. VK_KHR_ray_tracing_pipeline. Vulkan API reference. https://docs.vulkan.org/refpages/latest/refpages/source/VK_KHR_ray_tracing_pipeline.html Accessed 22 September 2026.
- [7] Nathan Morrical, Jonathan Tremblay, Yunzhi Lin, Stephen Tyree, Stan Birchfield, Valerio Pascucci, and Ingo Wald. 2021. NVISII: A Scriptable Tool for Photorealistic Image Generation. arXiv:2105.13962. <https://arxiv.org/abs/2105.13962>
- [8] NVIDIA. 2026. NVIDIA OptiX Ray Tracing Engine. Developer documentation. <https://developer.nvidia.com/rtx/ray-tracing/optix> Accessed 22 September 2026.
- [9] NVIDIA. 2026. vk_gltf_renderer: Vulkan RTX glTF Scene Renderer. Source repository. https://github.com/nvpro-samples/vk_gltf_renderer Accessed 22 September 2026. Upstream repository reference; the Reaigen glTF fork requires its own revision identifier.
- [10] OpenEXR Project. 2026. Technical Introduction to OpenEXR. Technical documentation. <https://openexr.com/en/latest/TechnicalIntroduction.html> Accessed 22 September 2026.
- [11] Reaigen s.r.o. 2026. Spatial Dataset Control: Production Telemetry and Scene Review Evidence. Internal monitoring and review captures, 21 September 2026. Dashboard snapshot at 21:22:21 CEST; associated timing charts and RGB-depth review captures.
- [12] Reaigen s.r.o. 2026. Spatial Dataset Creation: Control-Plane Source and Engineering Records. Internal source archive, 20 September 2026. Snapshot de47a255f935cf711d839494d5cb14e82a30ddce. Includes GLTF_RENDER_LANE.md and the 19 September evaluation. Renderer fork maintained separately.
- [13] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. 2021. Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. <https://arxiv.org/abs/2011.02523>
- [14] RunPod. 2026. Manage Pods: GraphQL API. Official provider documentation. <https://docs.runpod.io/sdks/graphql/manage-pods> Accessed 22 September 2026. GPU types, pricing, availability, and pod lifecycle operations.
- [15] RunPod. 2026. Pods: Pricing. Official provider documentation. <https://docs.runpod.io/pods/pricing> Accessed 22 September 2026. Platform billing scope, not a source for historical experimental prices.