Supplementary Material for Gaussians on Fire: High-Frequency Reconstruction of Flames

Jakob Nazarenus[1], Dominik Michels[2], Wojtek Palubicki[3], Simin Kou[4], Fang-Lue Zhang[4], Sören Pirk[1], and Reinhard Koch[1]

[1] Kiel University, Germany {jna, sp, rk}@informatik.uni-kiel.de
[2] KAUST, Saudi Arabia dominik.michels@kaust.edu.sa
[3] Adam Mickiewicz University, Poland Wojciech.Palubicki@amu.edu.pl
[4] Victoria University of Wellington, New Zealand {simin.kou, fanglue.zhang}@vuw.edu.nz

This document provides additional material complementing the main paper. We include qualitative visualizations of reconstructed scene trajectories and our reconstruction pipeline, our source code, extended implementation details, a detailed description of the captured fire dataset, our rolling shutter implementation, comprehensive per-scene results, scene decomposition details, a spatial holdout validation, tests on other volumetric phenomena, a discussion on the linear transmittance model, and an overview of failure cases.

1 Fire Removal

The fire in our experiments is mostly emissive and raises the brightness of each image pixel where it is present. For a short video sequence with a static camera and the only motion being emissive flames, we can thus perform a minimum intensity projection over the time axis to remove the fire from the scene and reveal the background behind the flames. Fig. 1 demonstrates this procedure. As shown in this example, even for large flames this method removes most of the flames very effectively, enabling a robust reconstruction of the static background.

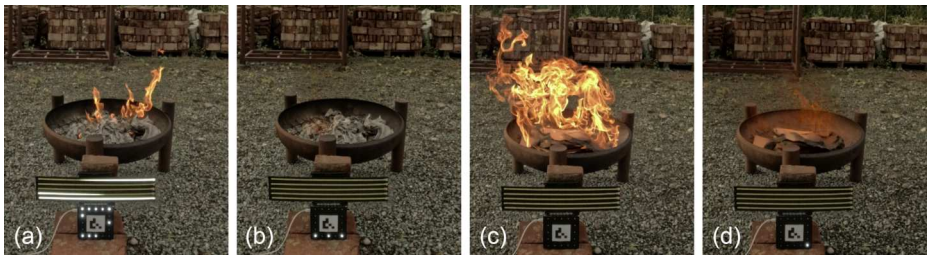

Fig. 1: Fire removal for dynamic scenes: we stack all frames along the sequence dimension and compute the minimum intensity projection over this axis to obtain an image of the static scene with most of the fire removed. We show a scene with a small (a,b) and larger flame (c,d) before and after this process.

2 Scene Decomposition

In Fig. 2, we visualize the different components of an exemplary scene. *Static* shows the result of the fire-removal process (with a small remaining fire artifact), *Dynamic* shows only the dynamic Gaussians, while *Combined* shows the rendered image of the full scene. The full-sequence videos in `videos/` contain per-frame dynamic-only renderings, which provide a more detailed view of the reconstructed flames across the whole sequence.

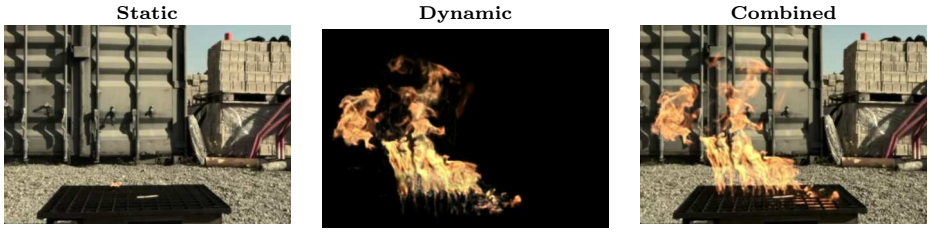

Fig. 2: Static/dynamic scene decomposition.

3 Rolling Shutter

Since most high-frame-rate consumer cameras rely on CMOS sensors with significant readout times, the resulting images are prone to rolling shutter distortion. In our experiments, we observed that for our pipeline, not explicitly handling rolling shutter distortions did not reduce the reconstruction quality in any metric. At the high frame rates, the inter-frame motion is comparatively small, which enables the reconstruction method to embed the rolling shutter distortion within the scene geometry, which removes the need for an explicit representation of rolling shutter effects.

However, to also handle scenes with much faster motion, we provide a simple extension to our method, which explicitly models rolling shutter effects. Given the sensor specifications and the undistortion parameters, we compute a non-linear delay function $t(\mathbf{p})$ for each image point $\mathbf{p}$. For a dynamic Gaussian observed at $\mathbf{p}_0$ at the start of the frame ($t = 0$), the projection of its linear worldspace motion itself is not linear, so we approximate its 2D motion as $\mathbf{p}(t) \approx \mathbf{p}_0 + t\,\mathbf{v}_{\text{image}}$, where $\mathbf{v}_{\text{image}}$ is the velocity in the image plane.

To estimate the temporal offset for this Gaussian, we locally approximate the nonlinear delay function as $t(\mathbf{p}) \approx t(\mathbf{p_0}) + \nabla t \cdot (\mathbf{p} - \mathbf{p}_0)$. Combining these two linear approximations, we find that the Gaussian is captured with a delay given by

$$t = \frac{t(\mathbf{p}_0)}{1 - \nabla t \cdot \mathbf{v}}\,. \tag{1}$$

For our setup, we observed a readout time of 2.85 µs, so for a maximum horizontal screen space motion of 50 px per frame, this gives $\nabla t \cdot \mathbf{v} \approx 5.5 \cdot 10^{-2}$. For such motions, we can approximate the temporal offset directly as $t(\mathbf{p}_0)$ in our rendering pipeline, yielding only sub-pixel approximation errors.

4 Dataset Details

To demonstrate the capabilities of our method, we need to show its results with respect to (1) the visual fidelity of the reconstruction, and (2) its physical plausibility. For the former we captured an extensive real-world dataset using three visually synchronized cameras, and for the latter, we ray-traced several simulated fire scenes and obtained their underlying velocity fields. This section gives an overview of the datasets created across both domains.

4.1 Real-World Dataset

For the validation of our method's visual reconstruction capabilities, we captured a diverse dataset of different fuels and backgrounds. We provide a more detailed overview of all captured scenes in Tab. 1. Each video has a duration of 15 s, a resolution of 1280×720, and a frame rate of 400 Hz. To capture the flames in their full height, all cameras were rotated by 90° to capture the scene in a portrait format. Fig. 3 shows our capturing setup as well as the visual synchronization hardware while Fig. 4 gives an overview of the captured videos.

Fig. 3: Real-world capturing setup. The left image depicts the camera and fire setup, while the right image shows the custom hardware used for visual synchronization of the cameras. The partially lit LEDs within its top section clearly depict the rolling shutter effect used for sub-frame synchronization.

4.2 Simulation

For the validation of our method's dynamic motion reconstruction capabilities, we created three separate *Mantaflow* fire simulations and ray-traced them in

Table 1: Summary of fire scenes, fuel types, and visual flame descriptions.

Scene	Fuel	Description
1	Propane	Low-intensity flame with mild wind from the left
2	Propane	High-intensity flame without wind; detached upper flames
3	Propane	Very low-intensity flame under quiescent conditions
4	Propane	Rapidly growing flame; pronounced vortical structures
5	Propane	Medium-height, wide flame exhibiting rapid growth
6	Propane	Sustained high flame with continuous combustion
7	Wood + Gasoline	Low-intensity flames propagating around wooden logs
8	Gasoline	Medium-scale flames with visible smoke production
9	Cardboard + Ethanol	Small, semi-transparent flames with visible smoke
10	Cardboard + Gasoline	Large flames under wind from the right
11	Cardboard	Small residual flames with visible ash formation
12	Cardboard + Gasoline	Large flames under wind from the right
13	Cardboard	Low-intensity residual flames with visible ashes
14	Paper + Gasoline	Very large flames under wind from the left
15	Paper	Small residual flames with visible ashes
16	Wood + Ethanol	Medium-scale flames; wood mostly unburnt
17	Wood + Ethanol	Small flames; wood partially charred

Blender. Similarly to the real-world dataset, we captured the scene using three rotated cameras with a resolution of 1280×720. For a feature-rich background, we chose to use the three public *classroom*, *junkshop*, and *monk* Blender scenes. We provide a qualitative overview of the rendered scenes in Fig. 5.

5 Per-Scene Results

In Tab. 2, we report the per-scene results of our model. The PSNR of the flame varies from 23.8 for scene 6 to 30.01 for scene 12. In Fig. 7, we provide additional qualitative results for multiple scenes. We also include per-scene full-sequence videos containing the ground truth, rendered RGB, and rendered depth in `videos/`.

6 Video Metrics

In our main paper, we provide per-frame metrics such as PSNR or SSIM to evaluate the visual fidelity of our reconstructed scenes for held-out test frames. Aside from per-frame metrics, there are several metrics that directly predict the perceptual differences between pairs of full videos. These provide deeper insights into temporal artifacts, like rapid changes in parts of the scene or flickering. We apply the Computer Graphics Video Quality Metric (CGVQM) [3] as a full sequence metric to compare each rendered scene with its ground truth data. Before processing, we softly masked out the synchronization pattern to focus

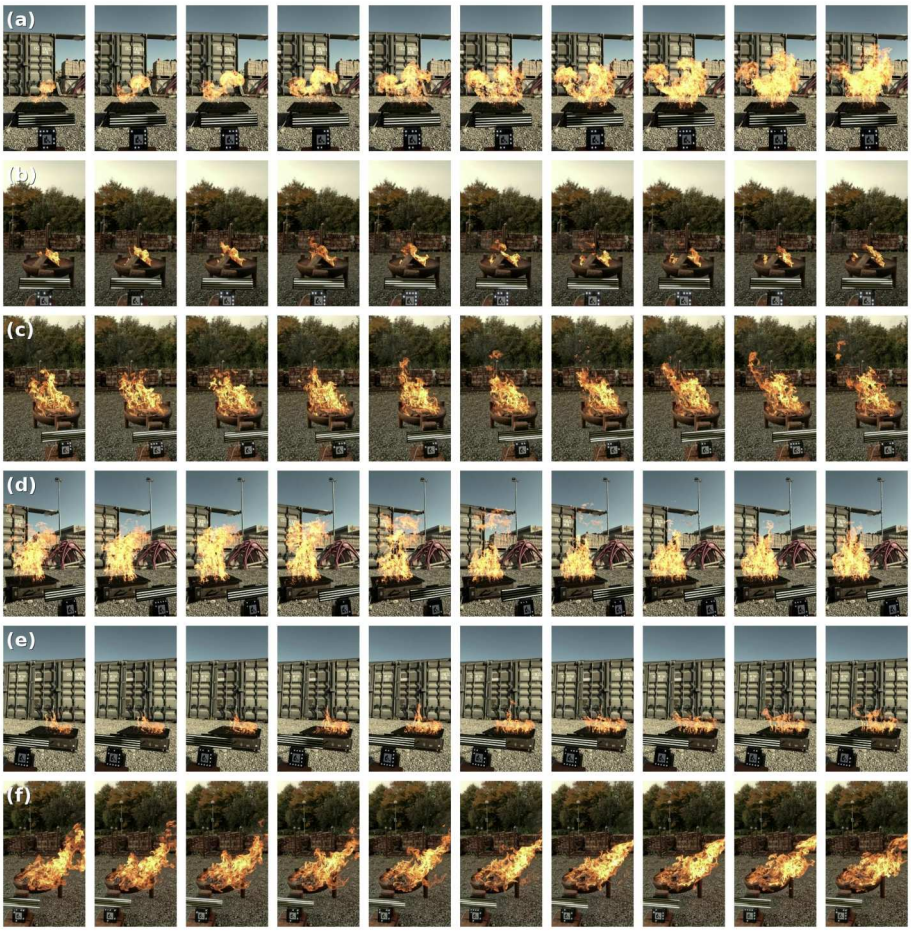

Fig. 4: Qualitative overview of our real-world fire dataset. Sequences (a,d,e) depict a propane jet of varying intensity, (b) is comprised of wooden logs, (c) are sheets of cardboard, and (f) is a pile of crumpled paper. Each scene progresses temporally from left to right.

on the dynamic fire and its static environment. Tab. 3 provides the quantitative results for this metric. Similarly to the per-frame results, our method significantly outperforms all baseline methods, with 4DGS [10] achieving second-best results.

7 Spatial Holdout Validation

While our method is primarily tailored towards physically plausible geometry (DVE) and temporally generalizable visuals, we can slightly adapt it to a spatial holdout setting. This is done by reducing the Gaussians' initial lifetime and increasing the lifetime's learning rate, which enables more spatial detail for individual temporal snapshots. We evaluated this using a 5-camera setup with three

Fig. 5: Qualitative overview of our synthetic fire dataset. Sequence (a) depicts the *junkshop*, (b) *classroom*, and (c) the *monk* scene. Each scene progresses temporally from left to right.

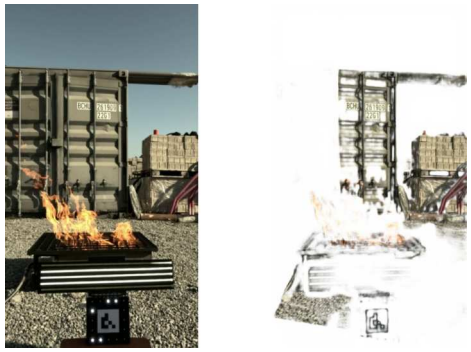

Fig. 6: Matched points using RoMa [1] for the real-world fire scene. The left image shows the view from the middle camera and the right image shows the predicted dense matches to the left neighboring camera. Even though the fire is fully observed from both views, RoMa only successfully matches parts of the flame.

cameras for training (as in our real-world experiments) and two additional intermediate cameras held out for validation. Tab. 4 shows the results for this setting. We additionally included two dynamic NeRF-based baselines, K-Planes [2] and MSTH [6]. While all Gaussian-based methods (including our method) degrade gracefully under this setting, the NeRF-based methods show a catastrophic failure due to their lack of a robust initialization.

Table 2: Per-scene evaluation results across all real-world scenes.

Scene	$\text{PSNR}_{\text{flame}}$ (↑)	$\text{SSIM}_{\text{flame}}$ (↑)	$\text{RMSE}_{\text{depth}}$ (↓)	LPIPS (↓)	PSNR (↑)	SSIM (↑)
1	26.67	0.847	0.089	0.027	35.96	0.977
2	26.16	0.810	0.097	0.041	31.91	0.950
3	28.61	0.892	0.101	0.014	37.64	0.982
4	24.99	0.762	0.122	0.048	33.04	0.957
5	26.72	0.795	0.133	0.030	34.78	0.967
6	23.80	0.765	0.083	0.059	30.14	0.942
7	25.56	0.822	0.077	0.034	36.59	0.972
8	24.98	0.805	0.084	0.042	34.35	0.962
9	26.96	0.864	0.109	0.037	37.29	0.971
10	26.48	0.837	0.208	0.048	33.75	0.952
11	28.55	0.891	0.074	0.025	39.38	0.976
12	30.01	0.878	0.095	0.029	37.98	0.972
13	29.95	0.911	0.036	0.025	39.42	0.978
14	28.37	0.871	0.090	0.042	35.11	0.960
15	23.80	0.799	0.084	0.031	37.31	0.972
16	28.59	0.906	0.058	0.030	38.30	0.973
17	27.69	0.873	0.055	0.026	37.10	0.973

Table 3: Comparison of CGVQM scores across different methods. The scores are aggregated over the scenes. Our method significantly outperforms all other baselines.

Method	CGVQM (↑)
Ours	86.1 ± 2.8
4DGS [10]	80.2 ± 5.0
Grid4D [9]	75.9 ± 5.3
4DGS [8]	73.5 ± 7.4
FreeTimeGS [7]	43.6 ± 26.7

Table 4: Spatial holdout evaluation on synthetic scenes.

Method	$\text{PSNR}_{\text{flame}}$ (↑)	$\text{SSIM}_{\text{flame}}$ (↑)	$\text{RMSE}_{\text{depth}}$ (↓)	LPIPS (↓)	PSNR (↑)	SSIM (↑)
Ours	20.54 ± 0.43	0.647 ± 0.018	0.190 ± 0.029	0.152 ± 0.016	20.89 ± 1.60	0.737 ± 0.054
FreeTimeGS	19.48 ± 0.41	0.590 ± 0.033	0.228 ± 0.018	0.346 ± 0.047	17.09 ± 0.76	0.590 ± 0.057
$\text{4DGS}_{\text{Yang}}$	19.05 ± 0.67	0.577 ± 0.036	0.286 ± 0.028	0.243 ± 0.025	19.08 ± 2.23	0.652 ± 0.057
4DGS_{Wu}	17.94 ± 0.38	0.529 ± 0.031	0.224 ± 0.036	0.287 ± 0.050	19.33 ± 2.27	0.620 ± 0.115
Grid4D	17.72 ± 0.52	0.520 ± 0.035	0.244 ± 0.053	0.284 ± 0.040	18.45 ± 1.76	0.578 ± 0.096
MSTH	10.83 ± 5.45	0.351 ± 0.136	0.250 ± 0.046	0.505 ± 0.122	12.52 ± 1.84	0.407 ± 0.098
K-Planes	8.81 ± 1.01	0.344 ± 0.050	0.289 ± 0.024	0.489 ± 0.027	12.11 ± 0.88	0.451 ± 0.082

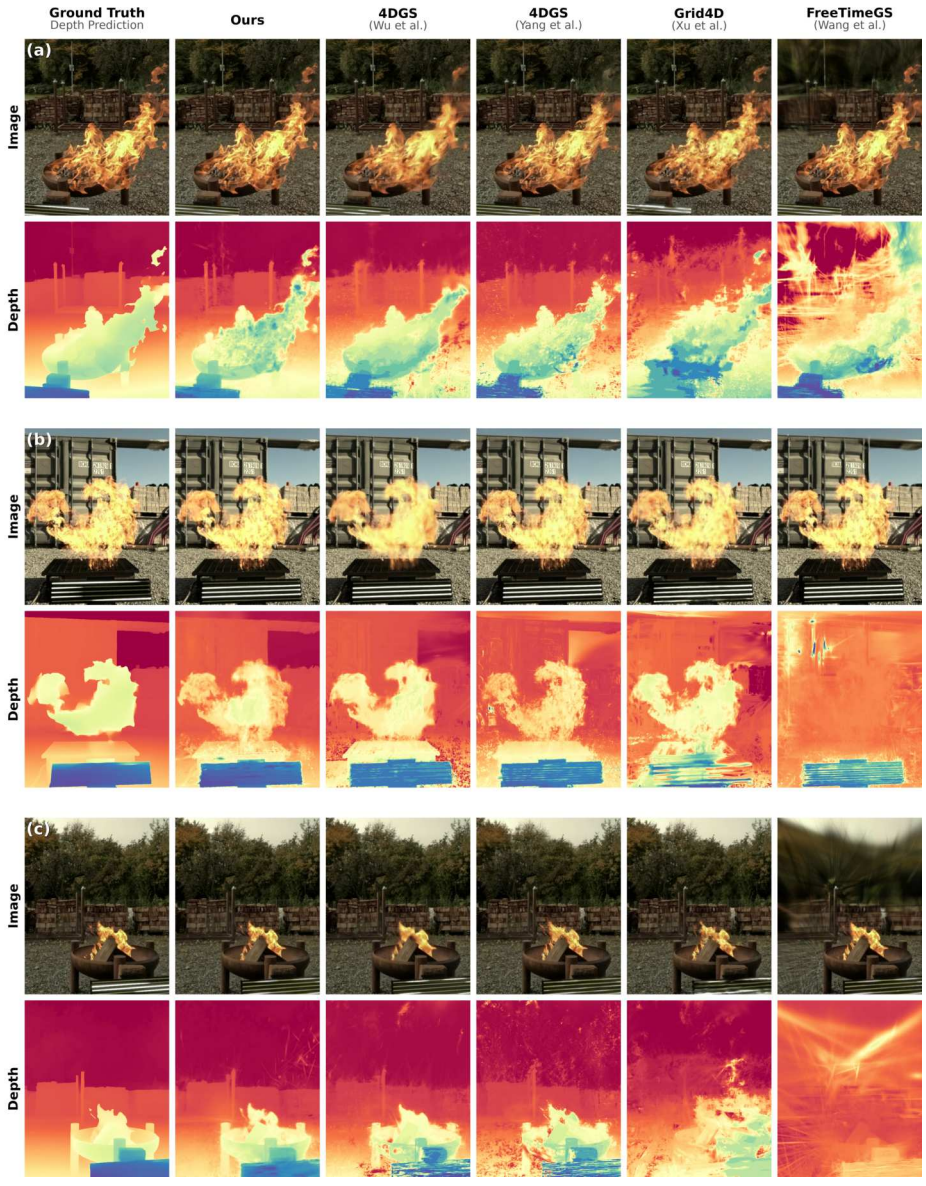

Fig. 7: Qualitative reconstruction results for multiple real-world scenes. Aside from our proposed method, all baselines suffer from visual or depth degradation.

8 Other Domains

While our method is mainly built around reconstructing fire, the underlying pipeline is in principle also applicable to other dynamic phenomena. To test the limits of this, we rendered three additional synthetic scenes containing smoke, a waterfall, and rain. Fig. 8 displays the qualitative results for these scenes, while Tab. 5 holds the quantitative results. The smoke scene is much less difficult to reconstruct visually due to the lack of detailed textures, while being more difficult to reconstruct volumetrically due to the inner turbulence and the lack of strong visual cues. For the water phenomena (waterfall and rain), we notice an opposite pattern: the visual reconstruction significantly degrades due to the non-linear trajectory of the light, while the volumetric reconstruction is more accurate due to its mostly linear gravity-dominated motion. These results demonstrate that a dense flow-initialized Gaussian Splatting pipeline can generalize to other volumetric dynamic phenomena, while also highlighting the overall limitations of Gaussian Splatting for reconstructing phenomena with complex light transport or non-linear motion. We included full-sequence videos for these scenes in `videos/`.

Table 5: Quantitative results for the reconstruction of other dynamic phenomena. Smoke performs best visually. The water scenes degrade visually, while being more physically accurate due to their gravity-dominated linear motion.

Scene	$\mathbf{PSNR_{flame}}$ (↑)	$\mathbf{SSIM_{flame}}$ (↑)	$\mathbf{RMSE_{depth}}$ (↓)	**LPIPS** (↓)	**PSNR** (↑)	**SSIM** (↑)	**CosSim** (↑)	$\mathbf{L_2}$ (↓)
Smoke	38.49	0.977	0.124	0.012	43.87	0.992	0.733	0.877
Waterfall	18.87	0.333	0.187	0.197	29.63	0.825	0.980	0.510
Rain	19.87	0.703	0.108	0.014	37.02	0.991	0.964	0.295

9 Exponential Transmittance

In vanilla 3D Gaussian Splatting, the transmittance is computed as a front-to-back product of per-Gaussian transmittance. While being very efficient, this is only an approximation of Beer-Lambert transmittance, which models the physically accurate exponential attenuation of light within a medium. For our scenes, this linear approximation is highly accurate due to the properties of the flame Gaussians. In our scenes, we measured each ray passing through the fire to intersect ≈ 300 Gaussians on average. Further, these Gaussians have an average size of half the voxel size used at initialization. For a large number of small Gaussians, the linear transmittance approaches a faithful exponential transmittance. To experimentally validate this, we reconstructed three synthetic scenes for a single point in time, once with the vanilla linear transmittance and once with an official exponential transmittance rasterizer [5] and validated them using spatially held out views. The results in Tab. 6 and Fig. 9 show no significant difference between the two rasterizers, confirming our hypothesis.

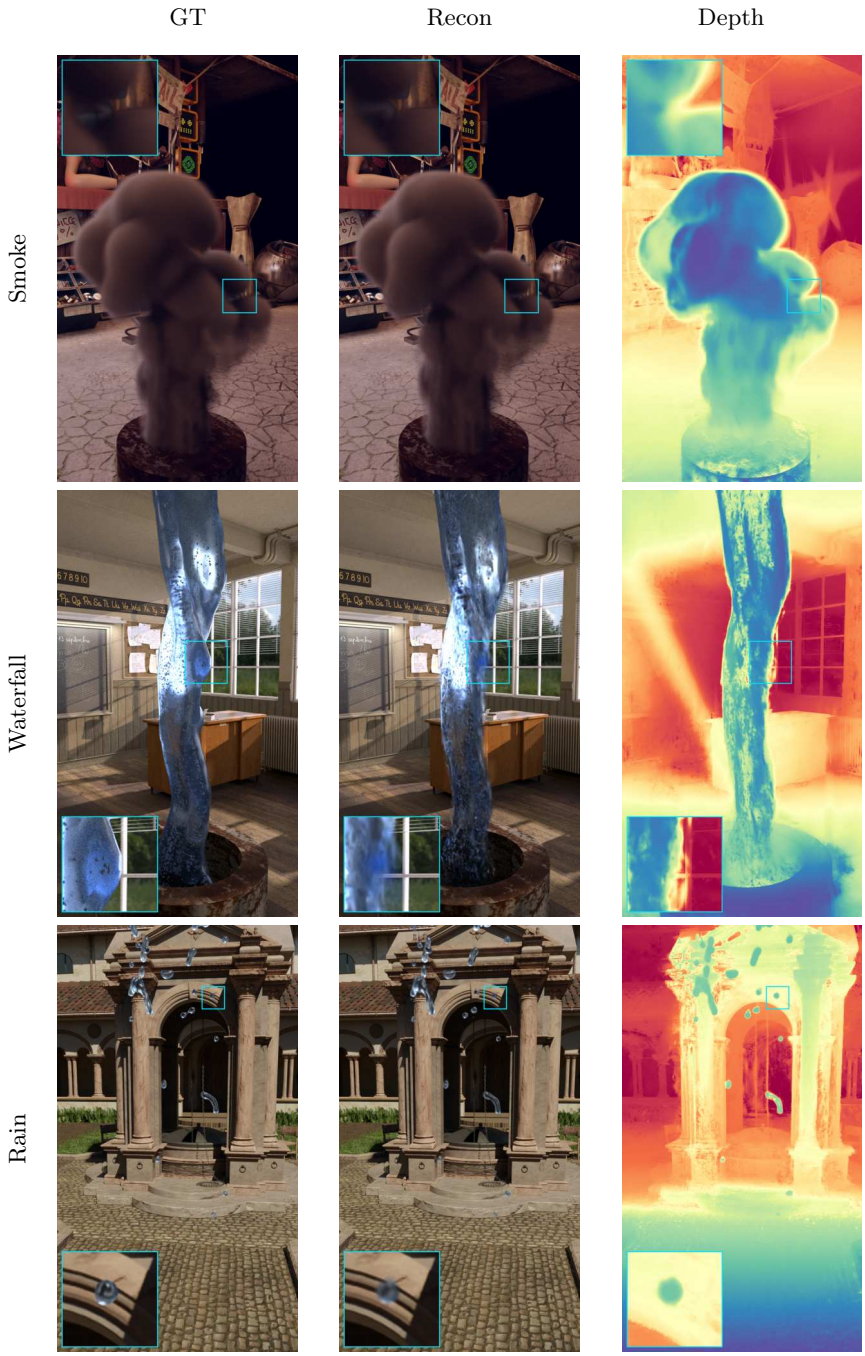

Fig. 8: Qualitative results for the reconstruction of other dynamic phenomena. The first column shows the ground truth images, the middle column shows the reconstruction's rendered color while the right column shows its rendered depth.

Fig. 9: Qualitative comparison of linear and exponential transmittance for three synthetic scenes. Aside from minor visual differences in the reconstruction artifacts, both transmittance models yield visually similar results. Depth is not included, as the reference implementation of the exponential transmittance rasterizer does not provide it as an output.

Table 6: Quantitative comparison of linear and exponential transmittance for three synthetic scenes. While the exponential transmittance model yields slightly degraded results, the differences are minor and do not significantly affect the visual reconstruction quality.

Transm.	$\mathbf{PSNR_{flame}}$ (↑)	$\mathbf{SSIM_{flame}}$ (↑)	**PSNR** (↑)	**SSIM** (↑)	**LPIPS** (↓)
Linear	18.51 ± 0.56	0.553 ± 0.032	20.46 ± 1.49	0.722 ± 0.055	0.170 ± 0.016
Exp.	18.16 ± 0.56	0.545 ± 0.034	20.20 ± 1.36	0.708 ± 0.053	0.197 ± 0.019

10 Failure Cases

Aside from the typical failure modes of Gaussian Splatting methods such as near-camera floaters (see Fig. 5 in the main paper), we observed two main failure modes that are specific to our proposed method. Both are shown in Fig. 10. The first failure mode is an incorrectly predicted 2D flow at initialization, omitting parts of the fire. These regions are not properly initialized with dynamic Gaussians, which leads to the absence of flames in the subsequent reconstruction. This optimization mostly cannot recover from this failure. The second failure mode is a blurred flame, which is a direct result of the linear motion assumption used in our whole pipeline. If this assumption is violated, the linear-motion approximation of non-linear motion leads to a smeared reconstruction. To observe these and other failure modes of our method, the full-sequence videos in `videos/` contain per-frame error maps for the rendered color.

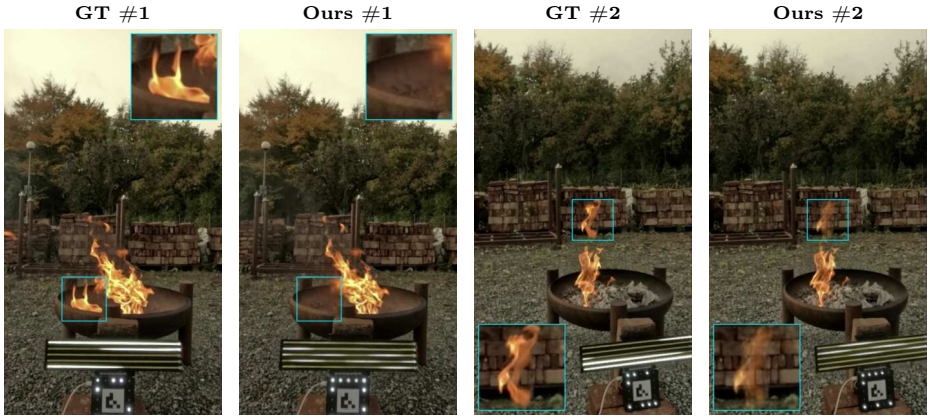

Fig. 10: Failure cases for two real-world scenes.

11 Novel View Trajectories

To provide a qualitative demonstration of our model's capability to generalize to novel views, we provide the video `results_and_pipeline.mp4`. In the video, we show the rendered results for different kinds of trajectories, leaving the camera poses used for training. For each timestep, we provide the rendered image as well as the rendered depth. To further visualize the dynamics of the reconstructed fire, we additionally render only the dynamic fire Gaussians at a reduced scale. This provides a view of the underlying motion field. For the comparison methods [7–10], we show their results along the same trajectories. All comparison methods yield significantly inferior reconstruction results, with blurry flames [8], flickering artifacts [9, 10], or deteriorated depth [7]. When focusing on the reconstructed motion field, it is clearly visible that while our method produces dense and plausible motion, all other baselines yield implausible Gaussian motion, with fire particles moving downwards or remaining stationary.

It is further necessary to demonstrate the graceful degradation of the reconstruction quality when completely leaving the proximity of the training views and entering highly novel views. To show this, we rendered a sequence of views along a circular trajectory around the fire, showing it from the side and from behind (see Fig. 11). As expected, this unmasks some artifacts in less-supervised regions of the scene, especially in Gaussians that are nearly fully occluded in the training views. However, the results clearly show that the reconstruction is not collapsing to a 2D billboard, but rather degrades gracefully while retaining the overall visual and volumetric plausibility of the scene.

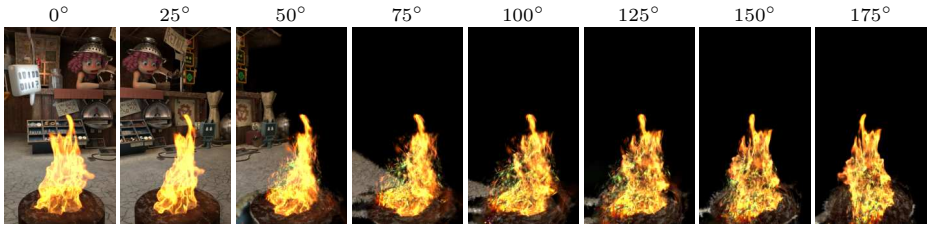

Fig. 11: Reconstruction of the synthetic fire scene for highly novel views. The camera rotates around the scene, leaving the proximity of the training views. While this unmasks some artifacts in the less-supervised regions, the overall reconstruction remains visually and volumetrically plausible. There is no visible 3D to 2D billboard collapse in the flames.

12 Pipeline Visualization

In the second half of the same video file we present a visualization of our proposed reconstruction pipeline. For one scene, this video shows the relevant preprocessing and optimization steps. For each stage, we show the corresponding view for

each of the three cameras. We start with the captured input videos and proceed with the process of removing the flames to obtain a static scene. Based on these images, we continue with the estimated depth maps and the Gaussian reconstruction of the static scene. For the dynamic scene, we first show the dense optical flow, followed by the optimization of the dynamic fire scene as the final stage.

13 Implementation Details

For the static scene optimization, we applied the original 3DGS implementation [4]. To account for the underconstrained scene geometry, we increased the depth loss by a factor of 100. Furthermore, to avoid the collapse of the scene geometry due to limited view constraints, we disabled the pruning during the densification process. We trained for 10^4 iterations with standard optimization parameters.

For the dynamic optimization, we modified the original 3DGS implementation [4] by adding velocity and lifetime parameters, similar to FreeTimeGS [7]. Each Gaussian additionally received a boolean parameter indicating whether it is to be considered a static background Gaussian or a dynamic fire Gaussian. During rendering, the temporal properties regarding position and opacity were purely applied to the Gaussians marked as dynamic. The optimization process including gradient-based optimization and densification was left as in the original implementation. To ensure a detailed reconstruction of the dynamic flames, we artificially limited the scale of the dynamic Gaussians by replacing their exponential with a sigmoid activation function, scaled by the voxel size used to initialize the scene. To avoid dynamic Gaussians representing parts of the static background that undergo slight temporal variations, we apply an additional *alpha loss* with a weight of 0.1. For this, we render the scene twice, first as the regular RGB scene, and secondly only the opacity of the dynamic Gaussians. Based on the rendered opacity and the pre-processed motion mask, we then compute the additional loss term. For a strong initialization of the motion field, we reduced the spatial learning rate by a factor of 10^{-2} for the trainable velocity parameter. We trained for $3 \cdot 10^4$ iterations with a batch size of 1, evaluating on the test set at iterations 1000, 7000, and 30000. For the further evaluation and comparison, we utilized the trained parameters at the final iteration.

Regarding the baseline methods [8–10], we relied on the unmodified implementations provided by the authors. Adhering to the official implementation, we initialized all methods on COLMAP points and trained with the optimization parameters provided by the authors for the *flame salmon* scene. As there is no official implementation for FreeTimeGS [7], we applied an unofficial implementation published on GitHub as `OpsiClear/FreeTimeGsVanilla`. Instead of initializing from COLMAP points, it computes pairwise RoMa [1] matches for each pair of simultaneous frames and initializes the motion using k-nearest-neighbors on temporally adjacent per-frame point clouds. Fig. 6 shows the issues with this initialization approach. While RoMa successfully predicts a dense model of the

static background scene, large sections of the fire are not densely matched or even incorrectly matched to parts of the static scene.

14 Code

We include the Python source code for preprocessing, as well as for the static and dynamic optimization, in the `code` directory. Due to size limitations of the supplementary material, we provide only a single short example scene in the `data` subdirectory. The source code contains README files that describe the execution workflow in detail. The full dataset and complete codebase will be released publicly.

15 Compute Hardware

For training, we used a system with a 16-core AMD Ryzen Threadripper PRO 3955, an NVIDIA RTX A6000 with 48 GB of GPU memory, as well as 128 GB of system memory. With this setup, training a single scene takes approximately 30 min at a GPU memory usage of 8 GB and a system memory usage of 2 GB. Rendering a single image has been measured to take approximately 7.4 ms, resulting in a frame rate of 131 Hz.

References

1. Edstedt, J., Sun, Q., Bökman, G., Wadenbäck, M., Felsberg, M.: RoMa: Robust Dense Feature Matching. In: CVPR (2024)
2. Fridovich-Keil, S., Meanti, G., Warbæk, F.R., Recht, B., Kanazawa, A.: K-planes: Explicit radiance fields in space, time, and appearance. In: CVPR (2023)
3. Jindal, A., Sadaka, N., Thomas, M.M., Sochenov, A., Kaplanyan, A.: CGVQM+D: Computer Graphics Video Quality Metric and Dataset. Computer Graphics Forum (2025)
4. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics (SIGGRAPH) **42**(4), 1–14 (2023)
5. Talegaonkar, C., Belhe, Y., Ramamoorthi, R., Antipa, N.: Volumetrically consistent 3d gaussian rasterization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10953–10963 (2025)
6. Wang, F., Chen, Z., Wang, G., Song, Y., Liu, H.: Masked space-time hash encoding for efficient dynamic scene reconstruction. Advances in neural information processing systems **36**, 70497–70510 (2023)
7. Wang, Y., Yang, P., Xu, Z., Sun, J., Zhang, Z., Yong, C., Bao, H., Peng, S., Zhou, X.: Freetimegs: Free gaussian primitives at anytime anywhere for dynamic scene reconstruction. In: CVPR (2025)
8. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Wang, X.: 4d gaussian splatting for real-time dynamic scene rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 20310–20320 (2024)

9. Xu, J., Fan, Z., Yang, J., Xie, J.: Grid4d: 4d decomposed hash encoding for high-fidelity dynamic gaussian splatting. In: Proceedings of the 38th International Conference on Neural Information Processing Systems. NIPS '24, Curran Associates Inc., Red Hook, NY, USA (2024)
10. Yang, Z., Yang, H., Pan, Z., Zhang, L.: Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In: ICLR (2024)