

# CamC2V: Context-aware Controllable Video Generation

## Supplementary Material

### 7. Computational Demand

Table 5 reports peak GPU memory and end-to-end inference latency for the baseline models and for our method. As shown, the *Context-aware Encoder* introduces only a small computational overhead relative to the base model. The visual stream exhibits a larger (yet still minor) latency than the semantic stream because it employs additional timestep- and pixel-wise query tokens to produce a dense visual representation.

Table 5. **Computational demand analysis.** Parameter counts for our method and the baselines, including a decomposition of our *context-aware encoder* into semantic and visual streams. Latency corresponds to the full generation process with 25 DDIM steps under classifier-free guidance (CFG). The encoder adds only minimal overhead and runs comfortably on consumer GPUs with < 16 GiB of VRAM.

| Method            | # Params  |        | GPU Memory (GiB) | Latency (s) |
|-------------------|-----------|--------|------------------|-------------|
|                   | Trainable | Total  |                  |             |
| DynamiCrafter     | -         | 2.6 B  | 10.43            | 5.143       |
| MotionCtrl        | -         | 2.6 B  | 10.49            | 4.849       |
| CameraCtrl        | -         | 2.8 B  | 11.28            | 4.898       |
| CamI2V            | -         | 2.9 B  | 11.21            | 7.976       |
| CamC2V            | 97.4 M    | 2.9 B  | 11.68            | 8.208       |
| – Semantic Stream | 50.9 M    | 50.9 M | 0.19             | 0.004       |
| – Visual Stream   | 46.5 M    | 46.5 M | 0.17             | 0.043       |

### 8. Camera Evaluation

We assess the quality of the generated camera trajectories by estimating camera poses with the GLOMAP pipeline. The non-default configuration used in our evaluation is listed in Table 7.

| Parameter                            | Value                                      |
|--------------------------------------|--|
| ImageReader.single_camera            | 1  |
| ImageReader.camera_model             | SIMPLE_PINHOLE                             |
| ImageReader.camera_params            | { <i>f</i> }, { <i>cx</i> }, { <i>cy</i> } |
| SiftExtraction.estimate_affine_shape | 1  |
| SiftExtraction.domain_size_pooling   | 1  |
| SiftMatching.guided_matching         | 1  |
| SiftMatching.max_num_matches         | 65536                                      |
| RelPoseEstimation.max_epipolar_error | 4  |
| BundleAdjustment.optimize_intrinsics | 0  |

Table 6. **GLOMAP Configuration.** Changed parameters of the Glomap pipeline used in our evaluation.

### 9. Additional Qualitative Results

To further assess the generative quality we provide further qualitative results of our method in Fig. 9 compared to the baseline method CamI2V.

### 10. COLMAP Evaluation

NeRF or 3D Gaussian Splatting-based approaches often rely on an initial sparse 3D reconstruction step to get an initial pointcloud of the scene. Typically, COLMAP [18] is used as a sparse 3D reconstruction pipeline. The pipeline can be split in three stages, namely *feature extraction*, *feature matching* and *point triangulation*. After the first stage, we insert the ground-truth camera poses into the pipeline to improve the triangulation performance and keep it fair to our model, which is also provided with ground-truth camera poses.

| Parameter                         | Value   |
|-----------------------------------|---|
| ImageReader.single_camera         | 1   |
| ImageReader.camera_model          | SIMPLE_PINHOLE  |
| ImageReader.camera_params         | { <i>f</i> }, { <i>f</i> <i>y</i> }, { <i>cx</i> }, { <i>cy</i> } |
| Mapper.tri_ignore_two_view_tracks | 0   |
| Mapper.num_threads                | 16  |
| Mapper.init_min_tri_angle         | 1   |
| Mapper.multiple_models            | 0   |
| Mapper.extract_colors             | 0   |

Table 7. **COLMAP Configuration.** Changed parameters of the COLMAP pipeline used in our evaluation.

### 11. Novel View Synthesis Qualitative Results

We provide additional qualitative results comparing our model against FrugalNeRF in Fig. 9. It is visible that FrugalNeRF not only suffers from an insufficient initial sparse reconstruction step, indicated by the black borders, but also introduces artifacts, especially for fine-grained details and edges, where the 3D representation is not sufficiently learned. Overall, our approach mitigates such artifacts and produces clearer more realistic looking images.

In Fig. 8, we show the success rate of this pipeline plotted against the distance between the two used frames. All failure cases come from the triangulation stage, where the model fails to confidently match and triangulate points between images. Especially, for larger frame distances the success rate significantly drops, even though the images

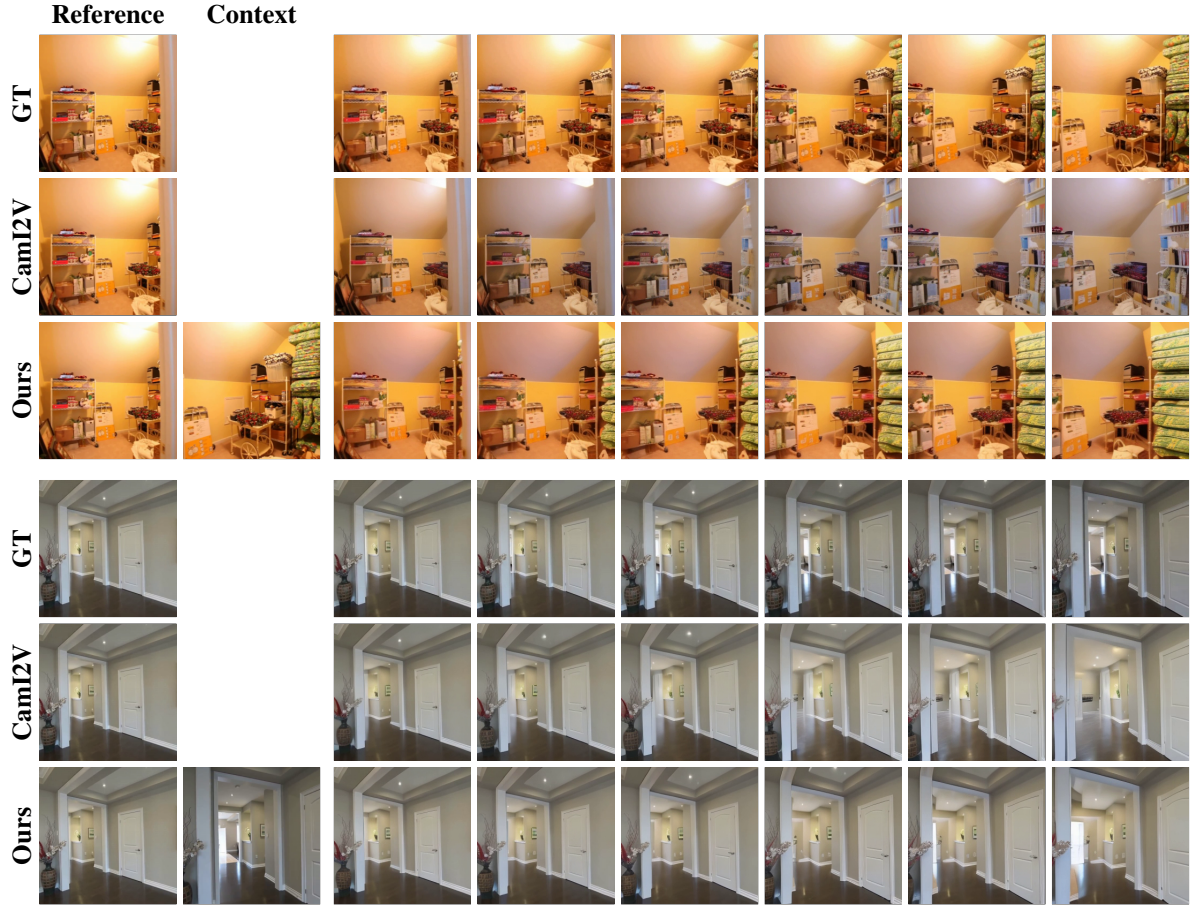


Figure 7. **Additional Qualitative results.** Zoom in for more details

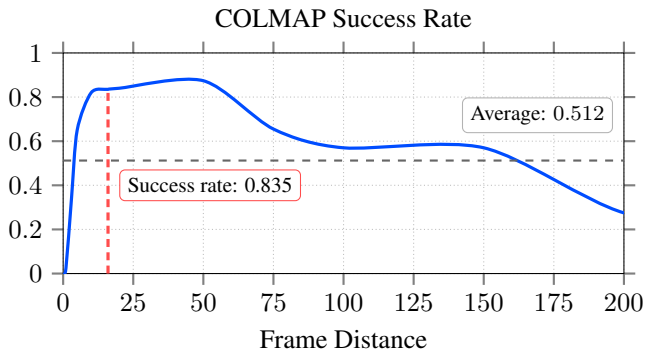


Figure 8. **COLMAP Triangulation.** We report the success rate of sparse two-view reconstruction in COLMAP across increasing inter-frame distances. This serves as the foundation of common NeRF and 3D Gaussian Splatting pipelines. The red dashed line indicates the COLMAP success rate for the setup used in our evaluation.

mately only leaves a small window from  $\sim 7$  to  $\sim 60$  where the pipeline is able to reliably produce a sparse 3D reconstruction in over 80% of the cases. Overall, in our used evaluation setup using the first and 17th frame the pipeline has a success rate of 83.5.

show sufficient overlap in our visual inspection. This ulti-



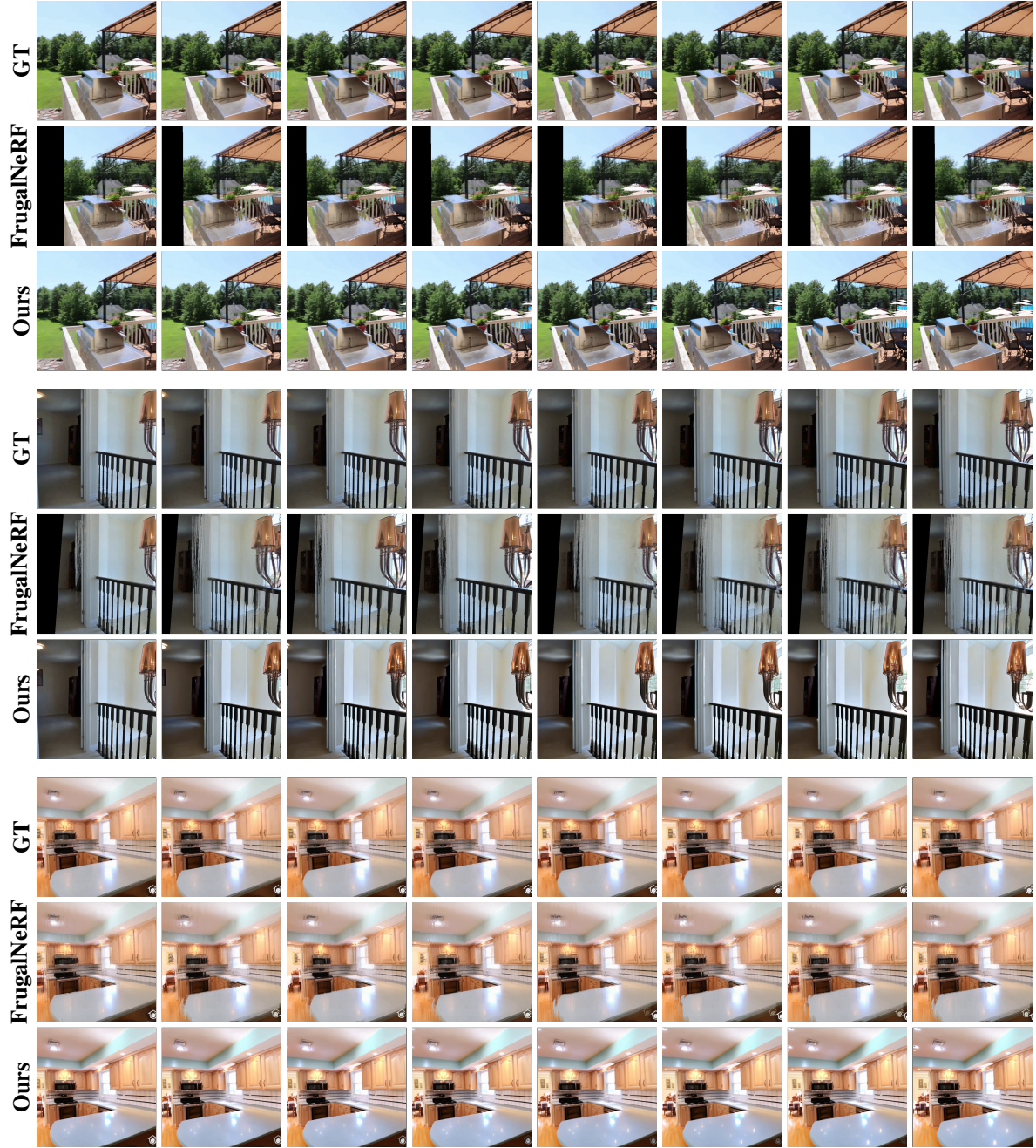


Figure 9. **Novel view synthesis qualitative results.** The images show the results from our model compared against the reconstructed scenes from FrugalNeRF. The coniditioning (or training frames for FrugalNeRF) are the first and last image.