
ProDyG: Progressive Dynamic Scene Reconstruction via Gaussian Splatting from Monocular Videos

—*Supplementary Material*—

Shi Chen
ETH Zürich

Erik Sandström
Google

Sandro Lombardi
Independent Researcher

Siyuan Li
ETH Zürich

Martin R. Oswald
University of Amsterdam

This supplementary material provides more details on the methodology and additional experimental results.

Contents

| | |
|--|----------|
| A Method | 1 |
| B More Experiments | 5 |
| B.1 Ablation Study | 5 |
| B.2 Runtime Evaluation | 6 |
| B.3 More Qualitative Results | 6 |

A Method

We describe further details about our method that were left out from the main paper.

Comparison to Existing Works. To further clarify the differences between ProDyG and the existing baseline methods, we classify each method in tab. S1 based on key characteristics. Our analysis shows that ProDyG is the first to simultaneously achieve online tracking and reconstruction, RGB-only input capability, global consistency, robust tracking against dynamic distractors, explicit dynamic scene reconstruction, and a 3DGS scene representation. Among the baseline methods, only DynaMoN [16], which uses a neural implicit scene representation, achieves all six characteristics. However, DynaMoN demonstrates inferior tracking performance compared to ProDyG, as shown in Table 1. Moreover, DynaMoN selects every eighth frame from the training sequences for the evaluation of novel view synthesis (NVS) performance on the Bonn [13] and TUM RGB-D [19] datasets, resulting in test views that are essentially very close to neighboring training views. Consequently, their novel view synthesis performance under larger viewpoint changes, as demonstrated in the iPhone dataset [3], remains unvalidated.

Semantic-guided Motion Mask Refinement. We describe the detailed algorithms for obtaining fine motion masks, separated into the initialization phase and the incremental prediction phase, as follows.

During the initialization phase (Algorithm 1), we first apply a 5×5 median filter to the coarse masks $\{C_i\}$ to remove salt-and-pepper noise. From these filtered masks, we extract connected regions with areas exceeding 5 pixels (in $8x$ downsampled resolution used by the SLAM backend [15]) and compute their centroids as candidate prompt points for SAM2. We then iterate through the following

| Method | Online Tracking | Online Reconstruction | RGB-only | Global Consistency | Robust Pose Estimation | Dynamic Reconstruction | Representation |
|-----------------------|-----------------|-----------------------|----------|--------------------|------------------------|------------------------|----------------|
| ORB-SLAM2 [12] | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | Point Cloud |
| NICE-SLAM [25] | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | Neural Impl. |
| ReFusion [13] | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | TSDf |
| DynaSLAM [1] | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | Point Cloud |
| DG-SLAM [23] | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | 3DGS |
| RoDyn-SLAM [4] | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | Neural Impl. |
| DDN-SLAM [7] | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | Neural Impl. |
| DSO [2] | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | Point Cloud |
| DROID-SLAM [21] | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | Point Cloud |
| MonoGS [11] | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 3DGS |
| Splat-SLAM [15] | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | 3DGS |
| MegaSaM [10] | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | Point Cloud |
| WildGS-SLAM [24] | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | 3DGS |
| DynaMoN [16] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | Neural Impl. |
| D4DGS-SLAM [20] | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | 3DGS |
| 4D-GS SLAM [9] | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | 3DGS |
| Shape of Motion [22] | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | 3DGS |
| DynOMo [17] | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ | 3DGS |
| MoSca [6] | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | 3DGS |
| Gaussian Marbles [18] | ✗ | ✗ | ✓ | ✓ | ✗ | ✓ | 3DGS |
| ProDyG (Ours) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 3DGS |

Table S1: **Method Classification.** We show that ProDyG is the first method to combine all characteristics including: online tracking and reconstruction, RGB-only input capability, global consistency, robust tracking against dynamic distractors, and explicit dynamic scene reconstruction and a 3DGS scene representation.

steps until no valid candidate prompt points remain or the number of accepted segmentations reaches a preset limit: 1) Prompt SAM2 at the point with the largest connected area in the candidate pool to segment the dynamic object across the sequence. 2) Validate the segmentation mask. A segmentation mask is accepted when it satisfies at least one of the criteria: a) Over 50 % of the keyframes where the segmented object appears contains at least one prompt point in the segmented area. b) The segmentation contains at least one prompt point in 3 consecutive keyframes. Otherwise, we reject it. 3) We zero out any pixels in the coarse motion masks that are identified in the predicted segmentation mask, and recompute prompt point candidates from the remaining connected regions after median filtering. Finally, we combine all accepted per-object segmentation masks using an OR operation to produce the final motion masks for all frames in the concerned sequence.

When a batch of new frames arrive, the incremental prediction pipeline (Algorithm 2) operates on a window spanning from $t_{-1} + 1$ through T , where t_0 is the timestamp of the last processed keyframe and t_{-1} points to the second-to-last processed keyframe. First, we adapt our flow threshold by computing a quantile threshold for the residual flow magnitude at t_0 , where the quantile level (q_{thresh}) is set as the ratio of dynamic pixels in the existing motion mask at t_0 . When nothing is segmented at t_0 (i.e. $q_{\text{thresh}} = 1$), we keep the threshold unchanged. This adaptive strategy automatically adjusts the sensitivity to motion by leveraging previous predictions, ensuring consistency across consecutive frame batches. After determining the adjusted threshold $\hat{\tau}$, we apply it to the residual flow magnitudes of the new keyframes to generate coarse motion masks. For each previously identified object, we prompt SAM2 to forward-propagate its segmentation to all new frames. We then zero out the pixels in the coarse masks that are inside these propagated segmentations. Following a median filtering step to remove noise, we proceed with the same iterative process as in the initialization phase.

Scale Factor Computation. When operating in RGB-D mode, our system must reconcile the arbitrary scale from the monocular SLAM backend with the metric scale provided by the depth sensor. To achieve this, we compute a robust global scale factor before each update of the static and dynamic Gaussians.

First, we convert the disparity estimates d from the SLAM backend to depths as $D_{\text{est}} = 1.0/d$. We then filter for valid depth values by applying the following criteria: (1) depth validity masks determined by multi-view consistency checks as described in [15], (2) depth range constraints on the estimated depths ($D_{\text{est}} > 10^{-3}$ and $D_{\text{est}} < 1000.0$), and (3) dynamic object masks to filter out the dynamic parts. For the ground truth depths, we consider them valid when $D_{\text{gt}} > 0$.

Algorithm 1: Motion Mask Initialization

Input: Frames $\{F_t\}_{t=1}^T$, keyframes timestamps $\mathcal{T} = \{t_i\}_{i=1}^{i_{\max}}$, residual flow magnitudes $\{\bar{r}_i\}_{i=1}^{i_{\max}}$
Output: Motion masks $\{M_t\}_{t=1}^T$

```
 $\{\tau_i\}_{i=1}^{i_{\max}} \leftarrow \text{ComputeQuantileThreshold}(\{\bar{r}_i\}_{i=1}^{i_{\max}}, 0.8);$   
 $\{C_i\}_{i=1}^{i_{\max}} \leftarrow \text{Threshold}(\{\bar{r}_i\}_{i=1}^{i_{\max}}, \{\tau_i\}_{i=1}^{i_{\max}});$  // Generate coarse masks  
 $n_{\text{obj}} \leftarrow 1;$  // Object index  
 $\text{valid\_indices} \leftarrow \emptyset;$  // Set of valid segmentation indices  
while true do  
   $\{C_i\}_{i=1}^{i_{\max}} \leftarrow \text{MedianFilter}(\{C_i\}_{i=1}^{i_{\max}});$  // Remove noise  
   $P \leftarrow \text{ExtractPromptCandidates}(\{C_i\}_{i=1}^{i_{\max}});$  // Extract centroids  
  if  $P = \emptyset$  or  $n_{\text{obj}} > n_{\max}$  then  
    break;  
   $p_{\text{best}} \leftarrow \text{LargestRegionCentroid}(P);$   
   $\{S_t^{n_{\text{obj}}}\}_{t=1}^T \leftarrow \text{SAM2}(p_{\text{best}}, \{F_t\}_{t=1}^T);$  // Segment object  
   $\text{ratio} \leftarrow \text{ValidateSegmentationRatio}(\{S_t^{n_{\text{obj}}}\}_{t \in \mathcal{T}}, P);$   
   $\text{consecutive} \leftarrow \text{CheckConsecutiveHits}(\{S_t^{n_{\text{obj}}}\}_{t \in \mathcal{T}}, P, 3);$   
  if  $\text{ratio} > 0.5$  or  $\text{consecutive} = \text{True}$  then  
     $\text{valid\_indices} \leftarrow \text{valid\_indices} \cup \{n_{\text{obj}}\};$  // Add to valid indices  
     $n_{\text{obj}} \leftarrow n_{\text{obj}} + 1;$  // Only increment if accepted  
   $\{C_i\}_{i=1}^{i_{\max}} \leftarrow \{C_i\}_{i=1}^{i_{\max}} \setminus \{S_t^{n_{\text{obj}}}\}_{t \in \mathcal{T}};$  // Zero out pixels  
 $\{M_t\}_{t=1}^T \leftarrow \text{CombineSegments}(\{\{S_t^j\}_{t=1}^T | j \in \text{valid\_indices}\});$  // OR operation  
return  $\{M_t\}_{t=1}^T$ 
```

For each keyframe i in the factor graph, we compute a frame-specific scale factor

$$\alpha_i = \frac{\sum_{p \in \mathcal{P}_{\text{valid}}^i} D_{\text{gt}}^i(p)}{\sum_{p \in \mathcal{P}_{\text{valid}}^i} D_{\text{est}}^i(p)}, \quad (1)$$

where $\mathcal{P}_{\text{valid}}^i$ represents the set of pixels in keyframe i where both estimated and ground truth depths are valid. To handle potential outliers among these keyframe-level scale factors, we employ an filtering procedure based on the Median Absolute Deviation (MAD):

$$\alpha_{\text{median}} = \text{median}(\{\alpha_i\}_{i=1}^{i_{\max}}), \quad \text{MAD} = \max(\text{median}(|\alpha_i - \alpha_{\text{median}}|), 0.02), \quad (2)$$

where we apply a minimum threshold to the MAD to prevent over-sensitivity to small variations. We then compute modified z-scores for each keyframe

$$z_i = \frac{|\alpha_i - \alpha_{\text{median}}|}{\text{MAD}/c}, \quad (3)$$

and classify keyframes as inliers when $|z_i| < 2.0$, where $c = 0.6745$ is the normalization factor that makes MAD-based z-scores equivalent to standard z-scores.

Finally, we aggregate all valid depth values from the inlier keyframes and compute the global scale factor as:

$$\alpha_{\text{global}} = \frac{\sum_{i \in \mathcal{I}} \sum_{p \in \mathcal{P}_{\text{valid}}^i} D_{\text{gt}}^i(p)}{\sum_{i \in \mathcal{I}} \sum_{p \in \mathcal{P}_{\text{valid}}^i} D_{\text{est}}^i(p)}, \quad (4)$$

where \mathcal{I} denotes the set of inlier keyframes. This robust approach ensures accurate scale alignment even in the presence of per-pixel depth errors and keyframe-level scale inconsistencies that might arise from dynamic objects, sensing errors, or tracking inaccuracies.

Image-Gradient-Based Gaussian Initialization. Unlike [6], which distributes Gaussians uniformly across all pixels, we employ an adaptive initialization strategy that varies the density of dynamic Gaussians according to the image gradient magnitude. Specifically, for each RGB frame I_t , we compute the gradient magnitude map G_t using Sobel operators:

$$G_t = \sqrt{(\nabla_x I_t)^2 + (\nabla_y I_t)^2}, \quad (5)$$

Algorithm 2: Incremental Motion Mask Prediction

Input: New frames $\{F_t\}_{t=t_{-1}+1}^T$, new keyframe timestamps $\mathcal{T} = \{t_i\}_{i=1}^{i_{\max}}$, previous keyframe timestamps $\{t_{-1}, t_0\}$, existing motion mask M_{t_0} , existing object-wise masks $\{S_{t_0}^n\}_{n=1}^{n_{\text{obj}}}$, residual flow magnitudes $\{\bar{r}_i\}_{i=0}^{i_{\max}}$, current adaptive threshold $\hat{\tau}$

Output: Updated motion masks $\{M_t\}_{t=1}^T$

```
qthresh = 1 - Mean( $M_{t_0}$ ) ; // Quantile for adaptive threshold
if qthresh < 1 then
     $\hat{\tau} \leftarrow \text{ComputeQuantileThreshold}(\bar{r}_0, q_{\text{thresh}})$  ; // Adapt threshold
 $\{C_i\}_{i=1}^{i_{\max}} \leftarrow \text{Threshold}(\{\bar{r}_i\}_{i=1}^{i_{\max}}, \hat{\tau}_{\text{flow}})$  ; // Generate coarse masks
for n = 1, 2, ..., nobj do
     $\{S_t^n\}_{t=t_0}^T \leftarrow \text{SAM2}(S_{t_0}^n, \{F_t\}_{t=t_0}^T)$  ; // Forward-propagate existing segmentation
     $\{C_i\}_{i=1}^{i_{\max}} \leftarrow \{C_i\}_{i=1}^{i_{\max}} \setminus \{S_t^n\}_{t \in \mathcal{T}}$  ; // Zero out pixels
nobj  $\leftarrow$  nobj + 1;
valid_indices  $\leftarrow$  {1, 2, ..., nobj} ; // Initialize with existing objects
while true do
     $\{C_i\}_{i=1}^{i_{\max}} \leftarrow \text{MedianFilter}(\{C_i\}_{i=1}^{i_{\max}})$  ; // Remove noise
    P  $\leftarrow$  ExtractPromptCandidates( $\{C_i\}_{i=1}^{i_{\max}}$ ) ; // Extract centroids
    if P =  $\emptyset$  or nobj > nmax then
        break;
    pbest  $\leftarrow$  LargestRegionCentroid(P);
     $\{S_t^{n_{\text{obj}}}\}_{t=t_{-1}+1}^T \leftarrow \text{SAM2}(p_{\text{best}}, \{F_t\}_{t=t_{-1}+1}^T)$  ; // Segment object
    ratio  $\leftarrow$  ValidateSegmentationRatio( $\{S_t^{n_{\text{obj}}}\}_{t \in \mathcal{T}}, P$ );
    consecutive  $\leftarrow$  CheckConsecutiveHits( $\{S_t^{n_{\text{obj}}}\}_{t \in \mathcal{T}}, P, 3$ );
    if ratio > 0.5 or consecutive = True then
        valid_indices  $\leftarrow$  valid_indices  $\cup$  {nobj} ; // Add to valid indices
        nobj  $\leftarrow$  nobj + 1 ; // Increment counter
     $\{C_i\}_{i=1}^{i_{\max}} \leftarrow \{C_i\}_{i=1}^{i_{\max}} \setminus \{S_t^{n_{\text{obj}}}\}_{t \in \mathcal{T}}$  ; // Zero out pixels
 $\{M_t\}_{t=1}^T \leftarrow \text{CombineSegments}(\{\{S_t^j\}_{t=1}^T | j \in \text{valid\_indices}\})$  ; // OR operation
return  $\{M_t\}_{t=1}^T$ 
```

where ∇_x and ∇_y denote the Sobel operators in the x and y directions, respectively. We then construct a probability distribution by normalizing the gradient magnitudes:

$$p_t(u, v) = \frac{G_t(u, v)}{\sum_{(u', v')} G_t(u', v')} , \quad (6)$$

where (u, v) are pixel coordinates. For dynamic Gaussian initialization, we sample from this distribution and thereby allocate more Gaussians to regions with higher gradient magnitudes. Additionally, we modulate the Gaussian scales inversely proportional to the local gradient magnitude:

$$\mathbf{s}_{(u, v)} = \mathbf{s}_{\text{base}} \cdot \frac{\bar{p}}{p_t(u, v)} , \quad (7)$$

where \mathbf{s}_{base} is the base scale computed from the depth as in [6], and \bar{p} is the average probability ($\bar{p} = 1/N$ for N pixels). We apply clamping to bound the modulated scale $\mathbf{s}_{(u, v)}$ within a range of $[\mathbf{s}_{\text{base}}/3, 3\mathbf{s}_{\text{base}}]$ to avoid extreme modulation. This approach enhances reconstruction efficiency and quality, as regions with rich textural detail receive more and smaller Gaussians to capture fine details, while homogeneous areas are modeled with fewer, larger Gaussians.

Final Refinement of Static Gaussians. Similar to Splat-SLAM [15], we perform a final refinement on the static Gaussians after the last global BA with an identical number of iterations. However, we extend this approach by additionally sampling a random non-keyframe for supervision during each iteration, but leave out the depth loss to prevent geometry inconsistency between keyframes and non-keyframes. Furthermore, we initialize new static Gaussians in regions not covered by the keyframes when first sampling a non-keyframe. These newly observed regions are defined as the

| Setup | PSNR \uparrow | SSIM \uparrow | LPIPS \downarrow |
|---------------------|-----------------|-----------------|--------------------|
| (a) Ours | 20.05 | 0.62 | 0.26 |
| (b) DynaGSLAM [8] | 19.18 | 0.51 | 0.31 |
| (c) Splat-SLAM [15] | 17.49 | 0.54 | 0.31 |

Table S2: **Ablation study of our motion mask prediction algorithm.** We report the NVS metrics tested on the paper-windmill sequence from the iPhone dataset [3], under three different setups: (a) ProdyG using the proposed motion mask prediction algorithm (Algorithm 1, 2); (b) ProDyG with the residual-flow-based prompting replaced by the approach applied in DynaGSLAM [8]; (c) Splat-SLAM [15] without any motion masking. ProDyG with our SAM2 [14] prompting algorithm (a) clearly outperforms the substitute method (b) and the case where no motion masking is applied (c) in terms of NVS quality.

| Metric | (a) Full Iterations | | | | | | (b) Reduced Iterations | | | | | |
|--------------------|---------------------|-------|-------|-------|-------|-------|------------------------|-------|-------|-------|-------|-------|
| | apple | block | paper | spin | teddy | Avg. | apple | block | paper | spin | teddy | Avg. |
| FPS \uparrow | 0.039 | 0.038 | 0.035 | 0.031 | 0.039 | 0.036 | 0.098 | 0.103 | 0.086 | 0.092 | 0.104 | 0.096 |
| Mem [GiB] | 20.65 | 20.14 | 14.05 | 19.79 | 20.33 | 18.99 | 16.40 | 16.60 | 10.04 | 16.41 | 14.64 | 14.81 |
| N_s [K] | 82.5 | 179.1 | 197.2 | 103.8 | 156.2 | 143.8 | 95.8 | 179.7 | 226.9 | 96.2 | 192.5 | 158.2 |
| N_d [K] | 5.7 | 10.3 | 3.9 | 7.1 | 24.2 | 10.2 | 3.3 | 5.5 | 3.0 | 2.5 | 17.9 | 6.4 |
| PSNR \uparrow | 18.93 | 16.17 | 20.62 | 18.87 | 14.65 | 17.85 | 19.17 | 15.45 | 20.48 | 18.03 | 14.70 | 17.57 |
| SSIM \uparrow | 0.773 | 0.590 | 0.656 | 0.670 | 0.577 | 0.653 | 0.776 | 0.590 | 0.650 | 0.656 | 0.574 | 0.649 |
| LPIPS \downarrow | 0.483 | 0.463 | 0.232 | 0.294 | 0.446 | 0.384 | 0.469 | 0.519 | 0.244 | 0.305 | 0.524 | 0.412 |

Table S3: **Runtime and Memory Evaluation on iPhone [3].** We report FPS, peak GPU memory usage, and the number of static and dynamic Gaussians alongside NVS metrics under two configurations: (a) “Full Iterations” uses the same setup as tab. 2 with 1500 geometry optimization iterations, 1600/8000 (online/final refinement) photometric optimization iterations, and 24576 initial CoTracker3 [5] tracks; (b) “Reduced iterations” uses 750 geometry optimization iterations, 400/2000 photometric optimization iterations, and 8192 initial tracks. ProDyG achieves approximately on average 0.1 FPS on the iPhone dataset when using reduced iterations with minimal performance degradation.

set of pixels p where the rendered opacity $O(p)$ (considering both static and dynamic Gaussians) falls below a threshold $\tau_{\text{opacity}} = 0.95$. This enhancement ensures comprehensive Gaussian coverage across all observed regions in the scene.

B More Experiments

In addition to the evaluations presented in the main paper, we provide further experiments in this section.

Implementation Details. Following the main paper, all supplemental experiments are also conducted with either an AMD EPYC 7H12 or 7742 CPU and an NVIDIA A6000 GPU. For all experiments of our method in the main paper, we run 1500 iterations of geometry optimization for each update of motion scaffolds, 1600 iterations of photometric optimization for each online update of dynamic Gaussians, and 8000 iterations of photometric optimization for a final refinement.

B.1 Ablation Study

To demonstrate the effectiveness of our algorithm design for motion mask prediction (Algorithm 1, 2), we show in tab. S2 an ablation study focused on NVS quality. Specifically, we compare the NVS performance on the paper-windmill sequence from the iPhone dataset [3] under three different setups: (a) ProDyG, using the proposed residual-flow-based-prompting of SAM2 [14]. (b) We replace the residual-flow-based prompting with a similar approach based on optical flow applied by DynaGSLAM [8]. Specifically, we compute and threshold the magnitude gradient of optical flow estimated by the SLAM backend to detect object edges, and close the shapes using dilations and erosions. After that, we prompt SAM2 at the centroids of connected regions in the resulting binary masks. (c) Splat-SLAM [15] where no motion masking is employed. ProDyG with our SAM2 prompting algorithm clearly outperforms the two ablated methods. The substitute method (b) tends to over-segment, mis-identifying some static foreground objects (e.g. the flower pot) as dynamic ones.

| Component | Total | Tracking | Geo. Opt. | Pho. Opt. | CoTracker3 [5] | Rest |
|-----------|-------|----------|-----------|-----------|----------------|------|
| Time [s] | 8052 | 22 | 1230 | 5424 | 785 | 591 |

Table S4: **Module-wise runtime breakdown** of ProDyG on the paper-windmill sequence from the iPhone dataset [3]. We split the total runtime into time for tracking, geometry optimization, photometric optimization, CoTracker3 [5] 2D point tracking, and the rest of the operations (*e.g.* data loading).

As a result, the static objects are reconstructed with some faulty motion, causing poor NVS quality in those regions. In comparison, our residual-flow-based method manages to decouple true scene motion from camera-induced motion and thus more accurate motion masks, resulting in superior NVS results.

B.2 Runtime Evaluation

We report our runtime and peak GPU memory usage evaluation on the iPhone Dataset [3] in tab. S3, and demonstrate that ProDyG can accelerate itself to approximately 0.1 FPS by reducing optimization iterations, with only a minimal performance loss. Under the reduced-iteration setup, our method outperforms the offline method Shape of Motion [22] in both NVS (PSNR and SSIM) and runtime. For instance, our total runtime on the 277-frame paper-windmill sequence was 53 minutes 40 seconds, which achieved PSNR = 20.48. In comparison, Shape of Motion took roughly 2 hours to finish using the same computing resources, while only scoring a PSNR = 18.14.

Additionally, we present in tab. S4 a module-wise runtime breakdown of ProDyG on the paper-windmill sequence from the iPhone dataset [3]. The geometry optimization of Motion Scaffolds [6], the photometric optimization of dynamic Gaussians, and CoTracker3 [5] tracking are the three main contributors of the runtime.

B.3 More Qualitative Results

In addition to the static qualitative results presented in the main paper, we show several supplementary qualitative results as images and videos on our [project page](#).

References

- [1] Bescos, B., F  cil, J.M., Civera, J., Neira, J.: Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE robotics and automation letters* **3**(4), 4076–4083 (2018) [2](#)
- [2] Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence* **40**(3), 611–625 (2017) [2](#)
- [3] Gao, H., Li, R., Tulsiani, S., Russell, B., Kanazawa, A.: Dynamic novel-view synthesis: A reality check. In: *NeurIPS* (2022) [1](#), [5](#), [6](#)
- [4] Jiang, H., Xu, Y., Li, K., Feng, J., Zhang, L.: Rodyn-slam: Robust dynamic dense rgb-d slam with neural radiance fields. *IEEE Robotics and Automation Letters* (2024) [2](#)
- [5] Karaev, N., Makarov, I., Wang, J., Neverova, N., Vedaldi, A., Rupperecht, C.: Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. *arXiv preprint arXiv:2410.11831* (2024) [5](#), [6](#)
- [6] Lei, J., Weng, Y., Harley, A., Guibas, L., Daniilidis, K.: Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. *arXiv preprint arXiv:2405.17421* (2024) [2](#), [3](#), [4](#), [6](#)
- [7] Li, M., Zhou, Y., Jiang, G., Deng, T., Wang, Y., Wang, H.: Ddn-slam: Real-time dense dynamic neural implicit slam. *arXiv preprint arXiv:2401.01545* (2024) [2](#)
- [8] Li, R.B., Shaghaghi, M., Suzuki, K., Liu, X., Moparthi, V., Du, B., Curtis, W., Renschler, M., Lee, K.M.B., Atanasov, N., et al.: Dynaslam: Real-time gaussian-splatting slam for online rendering, tracking, motion predictions of moving objects in dynamic scenes. *arXiv preprint arXiv:2503.11979* (2025) [5](#)
- [9] Li, Y., Fang, Y., Zhu, Z., Li, K., Ding, Y., Tombari, F.: 4d gaussian splatting slam. *arXiv preprint arXiv:2503.16710* (2025) [2](#)
- [10] Li, Z., Tucker, R., Cole, F., Wang, Q., Jin, L., Ye, V., Kanazawa, A., Holynski, A., Snavely, N.: Megasam: Accurate, fast, and robust structure and motion from casual dynamic videos. *arXiv preprint arXiv:2412.04463* (2024) [2](#)
- [11] Matsuki, H., Murai, R., Kelly, P.H., Davison, A.J.: Gaussian splatting slam. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 18039–18048 (2024) [2](#)
- [12] Mur-Artal, R., Tard  s, J.D.: Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics* **33**(5), 1255–1262 (2017) [2](#)
- [13] Palazzolo, E., Behley, J., Lottes, P., Giguere, P., Stachniss, C.: Refusion: 3d reconstruction in dynamic environments for rgb-d cameras exploiting residuals. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 7855–7862. *IEEE* (2019) [1](#), [2](#)
- [14] Ravi, N., Gabeur, V., Hu, Y.T., Hu, R., Ryali, C., Ma, T., Khedr, H., R  dle, R., Rolland, C., Gustafson, L., et al.: Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714* (2024) [5](#)
- [15] Sandstr  m, E., Tateno, K., Oechsle, M., Niemeyer, M., Van Gool, L., Oswald, M.R., Tombari, F.: Splat-slam: Globally optimized rgb-only slam with 3d gaussians. *arXiv preprint arXiv:2405.16544* (2024) [1](#), [2](#), [4](#), [5](#)
- [16] Schischka, N., Schieber, H., Karaoglu, M.A., Gorgulu, M., Gr  tzner, F., Ladikos, A., Navab, N., Roth, D., Busam, B.: Dynamon: Motion-aware fast and robust camera localization for dynamic neural radiance fields. *IEEE Robotics and Automation Letters* (2024) [1](#), [2](#)
- [17] Seidenschwarz, J., Zhou, Q., Duisterhof, B., Ramanan, D., Leal-Taix  , L.: Dynomo: Online point tracking by dynamic online monocular gaussian reconstruction. *arXiv preprint arXiv:2409.02104* (2024) [2](#)
- [18] Stearns, C., Harley, A., Uy, M., Dubost, F., Tombari, F., Wetzstein, G., Guibas, L.: Dynamic gaussian marbles for novel view synthesis of casual monocular videos. In: *SIGGRAPH Asia 2024 Conference Papers*. pp. 1–11 (2024) [2](#)
- [19] Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of rgb-d slam systems. In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)* (Oct 2012) [1](#)
- [20] Sun, Z., Lo, J., Hu, J.: Embracing dynamics: Dynamics-aware 4d gaussian splatting slam. *arXiv preprint arXiv:2504.04844* (2025) [2](#)
- [21] Teed, Z., Deng, J.: Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems* **34**, 16558–16569 (2021) [2](#)

- [22] Wang, Q., Ye, V., Gao, H., Austin, J., Li, Z., Kanazawa, A.: Shape of motion: 4d reconstruction from a single video. arXiv preprint arXiv:2407.13764 (2024) [2](#), [6](#)
- [23] Xu, Y., Jiang, H., Xiao, Z., Feng, J., Zhang, L.: Dg-slam: Robust dynamic gaussian splatting slam with hybrid pose optimization. arXiv preprint arXiv:2411.08373 (2024) [2](#)
- [24] Zheng, J., Zhu, Z., Bieri, V., Pollefeys, M., Peng, S., Armeni, I.: Wildgs-slam: Monocular gaussian splatting slam in dynamic environments. arXiv preprint arXiv:2504.03886 (2025) [2](#)
- [25] Zhu, Z., Peng, S., Larsson, V., Xu, W., Bao, H., Cui, Z., Oswald, M.R., Pollefeys, M.: Nice-slam: Neural implicit scalable encoding for slam. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12786–12796 (2022) [2](#)