ROAMing: Real-Time Occlusion-Aware 4D Surfel Mapping via Neural Scene Flow

David Recasens* NVIDIA, Zurich
David Tingdahl* NVIDIA, Zurich
Remo Steiner NVIDIA, Zurich
Vikram Ramasamy NVIDIA, Zurich
Alexander Millane NVIDIA, Zurich
Clemens Volk NVIDIA, Zurich
Tuur Stuyck NVIDIA, Santa Clara
Soha Pouya NVIDIA, Santa Clara

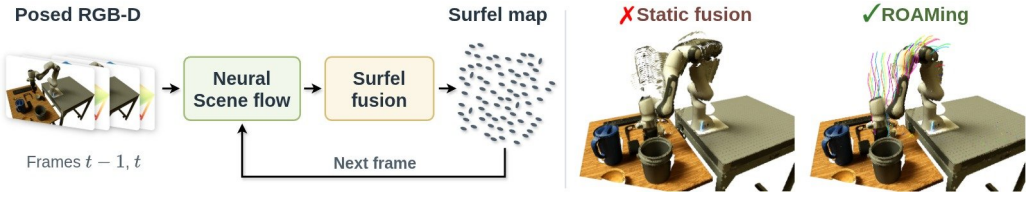


Figure 1: **ROAMing: warp before fusion for online dynamic mapping. Left:** Neural scene flow propagates the map from posed RGB-D frames before depth fusion. **Right:** Static fusion leaves stale arm geometry; ROAMing maintains a coherent map. Colored curves show 3D point tracks in an *ArenaPickPlace* test scene.

Abstract: Robots need maps that remain consistent through motion and occlusion. We present ROAMing, a causal RGB-D mapper that maintains an adaptive surfel map without object segmentation. A local-relative transformer refines lifted optical flow and propagates motion to non-visible neighbors; depth fusion adds newly observed surfaces and removes stale geometry. We introduce *ArenaPick-Place*, a synthetic robot-interaction dataset with complete surface trajectories. Our fast configuration achieves 88.3% completion of currently hidden surfaces on objects that move during the sequence. Evaluation includes zero-shot Sim4D reconstruction and real-world PW-DROID tracking. ROAMing trades some tracking accuracy relative to non-causal trackers for immediate updates. With precomputed optical flow, fast tracking and fusion take 56 ms per *ArenaPickPlace* frame.

Keywords: robotics, 3D perception, computer vision, motion estimation

1 Introduction

Robotic interaction requires a geometric map that remains useful when an arm or manipulated object moves out of sight. Static fusion leaves stale geometry (Fig. 1), while discarding moving measurements removes the objects of interest. Tracking a fixed set of visible points also misses surfaces revealed later. The challenge is to retain previously observed geometry through partial occlusion while incorporating newly visible surfaces as the scene changes.

ROAMing maintains an adaptive surface map by refining visible motion estimates and propagating motion to neighboring non-visible surfels before depth fusion. It operates on posed RGB-D frame pairs without object masks or future frames. Its contributions are: (i) a visibility-aware local-relative transformer that refines and spatially completes per-surfel 3D motion; (ii) its integration with standard surfel fusion to maintain a growing and shrinking dynamic map; and (iii) *ArenaPickPlace*, a ray-traced robot-interaction dataset with complete surface trajectories and visibility annotations for training and evaluation, including points that become occluded or move outside the camera view.

*Equal Contribution.

10th Conference on Robot Learning (CoRL 2026), Austin TX, USA.

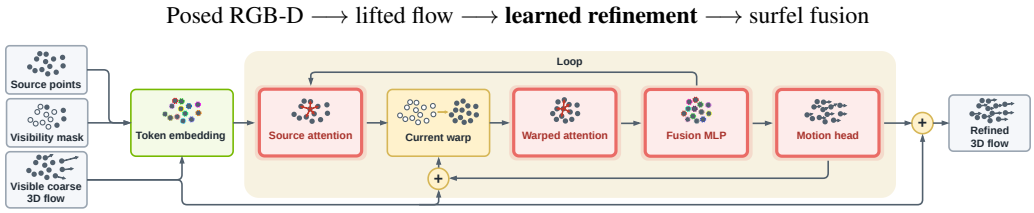


Figure 2: **Motion refinement before fusion.** Source attention uses a fixed neighborhood graph; warped attention recomputes neighbors as the motion estimate changes. Three refinement iterations predict a residual for every surfel, including those with no visible correspondence.

Related work. Dynamic SLAM can reject moving measurements to protect a static map [1]; CoFusion and MaskFusion instead reconstruct moving objects using instance segmentation [2, 3]. Among Gaussian surface mappers, 4DTAM [4] estimates a non-rigid warp online, and DynaGSLAM [5] combines flow with object masks. Track4World [6] and TAPIP3D [7] track fixed queries using full videos or overlapping windows. ROAMing couples causal motion prediction for visible and hidden surfels with incremental fusion. Further context is in App. A.

2 Method

At time t, ROAMing receives color I_t, depth D_t, intrinsics K_t, and camera pose $\mathbf{T}_{W \leftarrow C_t}$. Its map $\mathcal{M}_t = \{\mathcal{S}_t^i\}_{i=1}^{N_t}$ comprises N_t surfels [8] $\mathcal{S}_t$ with world-frame centers $\mathbf{p}_t^i$, normals, and radii. Each update first predicts motion for the existing map and then fuses the current depth image. The geometric prior supplies measured motion, the learned refiner corrects and propagates it (Fig. 2), and the fuser integrates observations and manages surfel creation and removal.

2.1 Neural Scene Flow

For every surfel, a geometric displacement $\mathbf{g}_t^i$ and learned residual $\mathbf{r}_t^i$ give the propagated center

$$\widetilde{\mathbf{p}}_t^i = \mathbf{p}_{t-1}^i + \mathbf{g}_t^i + \mathbf{r}_t^i = \mathbf{p}_{t-1}^i + \boldsymbol{\phi}_t^i. \tag{1}$$

Geometric prior. We lift forward optical flow from I_{t-1} to I_t into 3D using depth and the camera poses [9, 10]. Projecting $\mathbf{p}_{t-1}^i$ into the source image and following its flow gives a target pixel; back-projecting its depth into the world frame and subtracting $\mathbf{p}_{t-1}^i$ gives $\mathbf{g}_t^i$. This supplies motion estimates where valid correspondences exist. Hidden surfels and rejected flow/depth correspondences receive $\mathbf{g}_t^i = \mathbf{0}$, so this prior alone cannot move hidden geometry. The exact lift and validity checks are in App. B.

Visibility-aware refinement. The refiner (Fig. 2) predicts $\mathbf{R}_t = \mathcal{R}_\theta(\mathbf{P}_{t-1}, \mathbf{G}_t, \mathbf{v}_{t-1})$, where $\mathbf{P}_{t-1}$ stacks surfel centers, $\mathbf{G}_t$ their priors, and $\mathbf{v}_{t-1}$ their binary source visibility. Tokens encode the prior, its magnitude, visibility, and k-nearest-neighbor distance. Following Point Transformer [11], attention encodes relative spatial offsets rather than absolute positions.

Starting from zero residuals, we perform three iterations, inspired by RAFT's iterative refinement [12]. Each iteration first attends over a fixed source k-nearest-neighbor graph ($k_s = 16$), retaining source-surface context. It then warps the centers by the current $\boldsymbol{\phi}_t^i$, rebuilds a second graph ($k_w = 32$), and attends over these updated spatial neighbors. The source and warped attention stacks have distinct weights, each shared across iterations. Step-specific MLPs update the features, and motion heads predict residuals relative to the source-neighborhood centroid as in PointFlowNet [13].

A hidden surfel remains a token even when its prior is zero, allowing visible neighbors' motion features to influence its residual. Thus motion can propagate through partially observed local structure without object segmentation. To suppress accumulated drift in static regions, we set $\boldsymbol{\phi}_t^i = 0$ when $\|\mathbf{g}_t^i + \mathbf{r}_t^i\|_2 < 2\,\mathrm{mm}$. App. B gives the full architecture.

Table 1: **Dynamic reconstruction.** (a) Accuracy/completion@3 cm (%, ↑); second lines give mean map size (Total) or category shares. (b) Overlap-only depth L1 (cm, ↓). DynaGSLAM uses oracle Arena masks; it and ROAMing use 4DTAM poses on Sim4D. Arena costs: Table 3.

(a) *ArenaPickPlace*

Method	Total	Static		Dynamic	
		Visible	Hidden	Visible	Hidden
DynaGSLAM	96.4/60.7	83.1/56.8	80.7/57.5	98.8/99.9	86.9/60.7
	30k pts	36.7%	1.1%	60.0%	1.0%
4DTAM	81.4/82.9	89.6/86.6	75.6/77.9	74.7/66.2	56.1/55.7
	12k pts	66.7%	2.6%	16.7%	16.7%
Ours (accurate)	**97.4/99.1**	97.5/**99.6**	**95.5/98.9**	**99.4/100.0**	**89.7/90.2**
	102k pts	75.5%	5.2%	15.7%	3.3%
Ours (fast)	96.9/98.7	**97.6**/99.3	**95.5/98.9**	98.1/**100.0**	86.0/88.3
	31k pts	64.5%	4.8%	23.9%	6.1%

(b) Sim4D

Sequence	Dyna-GSLAM	4DTAM	Ours (acc.)	Ours (fast)
Curtain	5.81	**0.83**	1.06	1.09
Flag	5.97	3.18	**2.85**	2.97
Mercedes	3.17	0.86	**0.69**	0.82
Mod veh.	9.37	1.37	**0.87**	1.14
Rhino	7.78	1.17	**0.98**	1.36
Shoe rack	5.13	**1.28**	1.48	1.62
Water drop	6.41	3.82	**2.77**	3.42
Wave toy	17.12	**1.82**	3.44	4.26
Mean	7.59	1.79	**1.77**	2.08

2.2 Surfel Fusion

We apply established projective surfel fusion [14, 15, 16] to the propagated map. An information filter combines depth-consistent associations; uncovered measurements spawn surfels, free-space contradictions prune them, and occluded/out-of-frustum surfels persist. The learned warp updates centers; normals are refreshed on re-observation. Fusion equations and lifecycle details are in App. I.1.

3 Experiments

Data and training. *ArenaPickPlace* contains 38 tabletop robot manipulation scenes, with 10 different episode initializations and 10 cameras, with 200-frame RGB-D sequences and optical flow, meshes, poses, and surface point and normal trajectories. We train on all episodes/views of 17 scenes and test on the first episode/camera of the remaining. Training uses one-step rollouts from ground-truth points and optical flow with an $\mathcal{L}_1$ motion loss; testing rolls out recursively with estimated flow. Both operating points share weights and mapping parameters: *accurate* uses full-resolution RGB-D and WAFT [17]; *fast* uses half-resolution RGB-D and RAFT [12] (details: Apps. C, G).

3.1 Dynamic Reconstruction

We compare with 4DTAM [4] and DynaGSLAM [5]. On *ArenaPickPlace*, accuracy (map→GT) and completion (GT→map) measure the fraction of points within 3 cm of the other set, scoring only ever-observed geometry. *Dynamic* labels objects that move at any scored frame as dynamic throughout; *hidden* includes occluded, back-facing, and out-of-view points (thresholds: App. G). Zero-shot *Sim4D* [4] evaluation measures held-out-view depth L1 where rendered and ground-truth depth are valid, assessing accuracy, not coverage. Apps. F and E detail the protocol and baseline adaptations. Scores are single-run estimates.

Table 1a reports overall and subset accuracy/completion. With 31k versus 30k primitives, fast ROAMing improves dynamic-hidden completion over DynaGSLAM (88.3% versus 60.7%), with accuracy 86.0% versus 86.9%. ROAMing tracks more dynamic-hidden primitives: 6.1% versus 1.0% of these maps. Accurate ROAMing reaches 89.7/90.2% on this subset with 102k primitives; Fig. 3 illustrates hidden-geometry retention. Across eight Sim4D sequences (Table 1b), accurate ROAMing and 4DTAM have similar mean depth errors (1.77 versus 1.79 cm).

3.2 3D Tracking

Tracking EPE3D metric preserves first-frame query identities, without depth-based fusion correction, spawning/pruning. *ArenaPickPlace* supplies 200-frame trajectories; real-world PW-DROID [18] provides 12-frame pseudo-ground-truth clips. We restore its omitted robot-arm annotations using provided meshes. Track4World [6] uses full videos and ground-truth scale alignment; TAPIP3D [7] uses 16-frame windows with stride 8. ROAMing only $t-1$ and t to update its map.

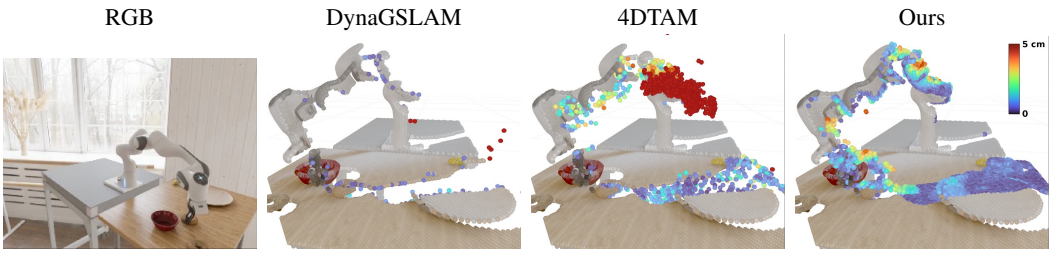


Figure 3: **Dynamic-hidden reconstruction** on *ArenaPickPlace* from a novel view. Target points show nearest-surface error (0–5 cm); geometry visible to the input camera supplies ground-truth RGB context.

Table 2: **Fixed-query 3D tracking.** Total EPE3D (cm, ↓), excluding spawning/pruning.

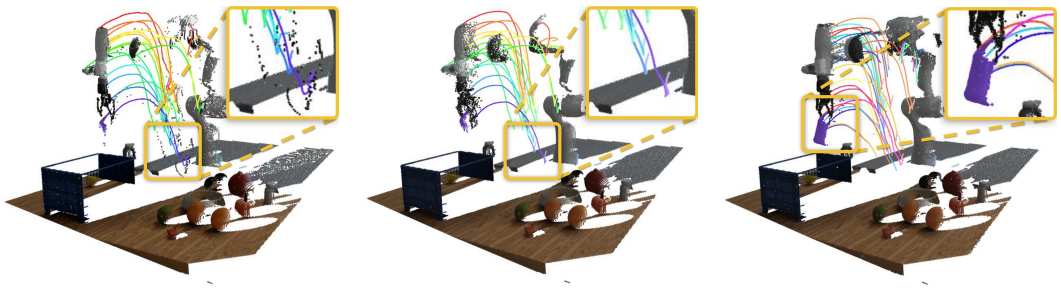
Method	*ArenaPickPlace*	PW-DROID
Track4World	13.61	12.21
TAPIP3D	**0.53**	**0.35**
Ours (accurate)	0.99	0.54
Ours (fast)	1.56	0.72

Table 3: **Runtime cost**. †: L40 GPU for methods exceeding the RTX 5090's 31.4 GiB.

	ArenaPickPlace		PW-DROID	
Method	ms/pair↓	Peak GiB↓	ms/pair↓	Peak GiB↓
Track4World	661†	35.6†	146	26.3
TAPIP3D	11,594†	42.6†	2,924	11.1
4DTAM	228	8.6	—	—
DynaGSLAM	149	**1.1**	—	—
Ours (accurate)	186	4.8	254	10.4
Ours (fast)	**56**	3.1	**58**	**1.9**

Table 2 reports Total fixed-query EPE: accurate ROAMing reaches 0.99/0.54 cm on Arena/PW-DROID versus TAPIP3D's 0.53/0.35 cm, trading accuracy for causal updates. Subset results are in Table SH.3 (Arena dynamic-hidden: 7.51 versus 4.26 cm). Table 3 reports runtime and VRAM: fast ROAMing uses 56/58 ms/pair and 3.1/1.9 GiB on Arena/PW-DROID, including fusion. Optical flow adds 14/10 ms (RAFT, fast) and 312/60 ms (WAFT, accurate). Clip inference is amortized.

Component analysis. Fig. 4 compares geometric flow, refinement, and fusion qualitatively.



(a) Geometric prior (b) With refinement (c) With fusion

Figure 4: **Component behavior.** Geometric flow leaves hidden points behind (a); refinement propagates them (b); fusion adds tracks on the disoccluded bottle (c). Sensor context and baselines: App. D, Fig. SD.1.

4 Conclusion and Limitations

ROAMing combines learned motion propagation through partial occlusion with standard fusion to maintain an adaptive surface map from causal observations in real-time, enabling robust, deployable 4D mapping for real robots under tight computational constraints. Reconstruction is evaluated only in simulation; training and the main benchmark use fixed cameras, leaving camera motion untested. The system depends on external poses and optical flow, and cannot infer arbitrary motion of fully hidden components disconnected from visible local structure. Real-world PW-DROID results evaluate 12-frame tracking clips, not the long-term accuracy and completeness of persistent maps.

References

[1] B. Bescos, J. M. Fácil, J. Civera, and J. Neira. Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE robotics and automation letters*, 3(4):4076–4083, 2018.

[2] M. Rünz and L. Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4471–4478. IEEE, 2017.

[3] M. Runz, M. Buffier, and L. Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 10–20. IEEE, 2018.

[4] H. Matsuki, G. Bae, and A. J. Davison. 4DTAM: Non-Rigid Tracking and Mapping via Dynamic Surface Gaussians, 2025. arXiv:2505.22859.

[5] R. B. Li, M. Shaghaghi, K. Suzuki, X. Liu, V. Moparthi, B. Du, W. Curtis, M. Renschler, K. M. B. Lee, N. Atanasov, et al. Dynagslam: Real-time gaussian-splatting slam for online rendering, tracking, motion predictions of moving objects in dynamic scenes. In *2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2434–2444. IEEE, 2026.

[6] J. Lu, J. Xu, W. Hu, R. Zhu, C. Zhao, S.-K. Yeung, Y. Shan, and Y. Liu. Track4world: Feedforward world-centric dense 3d tracking of all pixels. *arXiv preprint arXiv:2603.02573*, 2026.

[7] B. Zhang, L. Ke, A. Harley, and K. Fragkiadaki. Tapip3d: Tracking any point in persistent 3d geometry. *Advances in Neural Information Processing Systems*, 38:135284–135303, 2026.

[8] H. Pfister, M. Zwicker, J. Van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, 2000.

[9] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *IEEE International Conference on Computer Vision*, volume 2, pages 722–729, 1999. doi: 10.1109/ICCV.1999.790293.

[10] Y. Wang and G. H. Lee. Flow4DGS-SLAM: Optical flow-guided 4d gaussian splatting SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 33364–33373, 2026.

[11] H. Zhao, L. Jiang, J. Jia, P. H. S. Torr, and V. Koltun. Point transformer. In *IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.

[12] Z. Teed and J. Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[13] A. Behl, D. Paschalidou, S. Donné, and A. Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7954–7963. IEEE, 2019.

[14] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 1–8. IEEE, 2013.

[15] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense slam without a pose graph. In *Robotics: Science and Systems (RSS)*, 2015.

[16] X. Pan, Z. Li, Z. Ye, H. Zhai, and G. Zhang. Egg-fusion: Efficient 3d reconstruction with geometry-aware gaussian surfel on the fly. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–12, 2025.

[17] Y. Wang and J. Deng. Waft: Warping-alone field transforms for optical flow. In *International Conference on Learning Representations*, volume 2026, pages 157389–157402, 2026.

[18] W. Huang, Y.-W. Chao, A. Mousavian, M.-Y. Liu, D. Fox, K. Mo, and L. Fei-Fei. Point-world: Scaling 3d world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782*, 2026.

[19] K. Wang, F. Gao, and S. Shen. Real-time scalable dense surfel mapping. In *2019 International conference on robotics and automation (ICRA)*, pages 6919–6925. IEEE, 2019.

[20] N. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21357–21366. IEEE, 2024.

[21] Z. Peng, T. Shao, Y. Liu, J. Zhou, Y. Yang, J. Wang, and K. Zhou. Rtg-slam: Real-time 3d reconstruction at scale using gaussian splatting. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024.

[22] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger. Mid-fusion: Octree-based object-level multi-instance dynamic slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5231–5237. IEEE, 2019.

[23] Y. Wu and E. Demeester. Segfusion: Segmenting, tracking, and mapping of multiple unknown moving objects in dynamic slam. *Ieee Access*, 14:47453–47466, 2026.

[24] Y. Wang, J. Morris, L. Wu, T. Vidal-Calleja, and V. Ila. Dynorecon: Dynamic object reconstruction for navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3305–3311. IEEE, 2025.

[25] T. D. Ngo, A. Mirzaei, G. Qian, H. Liang, C. Gan, E. Kalogerakis, P. Wonka, and C. Wang. Deltav2: Accelerating dense 3d tracking. *arXiv preprint arXiv:2508.01170*, 2025.

[26] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. B. Girshick, P. Dollár, and C. Feichtenhofer. Sam 2: Segment anything in images and videos. In *ICLR*. OpenReview.net, 2025. URL `http://dblp.uni-trier.de/db/conf/iclr/iclr2025.html#RaviGHHR0KRRGMP25`.

[27] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys. NICE-SLAM: Neural implicit scalable encoding for SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[28] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, V. Guizilini, D. A. Herrera, M. Heo, K. Hsu, J. Hu, M. Z. Irshad, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2025. URL `https://arxiv.org/abs/2403.12945`.

[29] S. Koppula, I. Rocco, Y. Yang, J. Heyward, J. Carreira, A. Zisserman, G. Brostow, and C. Doersch. Tapvid-3d: A benchmark for tracking any point in 3d. *Advances in Neural Information Processing Systems*, 37:82149–82165, 2024.

[30] T. Weise, T. Wismer, B. Leibe, and L. Van Gool. In-hand scanning with online loop closure. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1630–1637. IEEE, 2009.

Supplementary Material

This supplement accompanies *ROAMing: Real-Time Occlusion-Aware 4D Surfel Mapping via Neural Scene Flow*. Citation numbers refer to the bibliography in the main paper.

A Additional Related Work

Surface representations. Volumetric TSDF maps provide explicit geometry but impose a spatial discretization. Point-based fusion instead represents an observed surface by oriented elements and supports efficient incremental updates [14, 15, 19]. Recent Gaussian SLAM systems emphasize differentiable rendering [20, 21]; EGG-Fusion [16] flattens Gaussian primitives into surfels with explicit normal tracking. ROAMing adopts surfels because the contribution is persistent motion estimation rather than a new static representation.

Dynamic reconstruction. Object-centric systems such as MID-Fusion [22], SegFusion [23], and DynORecon [24] reconstruct separate moving entities but depend on object masks or semantics. Flow4DGS-SLAM [10] and DynaGSLAM [5] use optical flow to update Gaussian maps, while 4DTAM [4] estimates a non-rigid warp field online. Long-range trackers [6, 7, 25] provide strong fixed-query trajectories but do not maintain an adaptive fused surface map and do not run in an online causal manner.

B Refinement Architecture

Geometric-prior validity. Let π denote pinhole projection and $\pi^{-1}(\mathbf{u}, d) = dK^{-1}\widetilde{\mathbf{u}}$ back-projection, with $\widetilde{\mathbf{u}}$ the homogeneous pixel coordinate. A source surfel projects to $\mathbf{u}^i_{t-1} = \pi(\mathbf{T}_{C_{t-1}\leftarrow W}\mathbf{p}^i_{t-1})$. Forward optical flow gives $\mathbf{u}^i_t = \mathbf{u}^i_{t-1} + \mathbf{f}_{t-1\to t}(\mathbf{u}^i_{t-1})$, and the lifted prior is

$$\mathbf{g}^i_t = \mathbf{T}_{W\leftarrow C_t}\pi^{-1}\big(\mathbf{u}^i_t, D_t(\mathbf{u}^i_t)\big) - \mathbf{p}^i_{t-1}. \tag{2}$$

For a surfel to receive a lifted prior, it must be visible in the source frame, the optical-flow sample must be valid and edge-safe, and the target pixel must have valid in-bounds depth. We reject surfels that are occluded or out of view, flow samples near depth discontinuities, and invalid flow/depth correspondences. A rejected correspondence receives $\mathbf{g}^i_t = \mathbf{0}$, but remains active in the learned refiner.

Tokens and neighborhoods. For each live surfel, the input token concatenates $\mathbf{g}^i_t$, $\|\mathbf{g}^i_t\|_2$, source visibility, and the distance to the farthest neighbor in the source and warped graphs. A linear layer embeds this token as $\mathbf{z}^i[0] \in \mathbb{R}^H$. Absolute positions are not embedded; local-relative attention encodes offsets. At each refinement step ℓ, a source-attention block uses the fixed k_s-nearest-neighbor graph. The current flow $\boldsymbol{\phi}^i_t[\ell] = \mathbf{g}^i_t + \mathbf{r}^i_t[\ell]$ then warps the centers, after which a second attention block uses a recomputed k_w graph. The two attention stacks have distinct parameters, shared across refinement steps.

Iterative update. Let $\widetilde{\mathbf{z}}^i[\ell]$ be the warped-attention output and $\mathbf{r}^i_t[0] = \mathbf{0}$. Each step updates

$$\mathbf{z}^i[\ell+1] = \widetilde{\mathbf{z}}^i[\ell] + \mathrm{MLP}_\ell\big([\widetilde{\mathbf{z}}^i[\ell], \mathbf{g}^i_t, \mathbf{r}^i_t[\ell], \boldsymbol{\phi}^i_t[\ell]]\big). \tag{3}$$

A step-specific motion head predicts the next residual displacement, expressed relative to the centroid of the fixed source neighborhood, following the translation-equivariant coordinates of PointFlowNet [13]. After L iterations, $\mathbf{r}^i_t = \mathbf{r}^i_t[L]$, and the norm threshold in Sec. 2.1 suppresses small static-region updates.

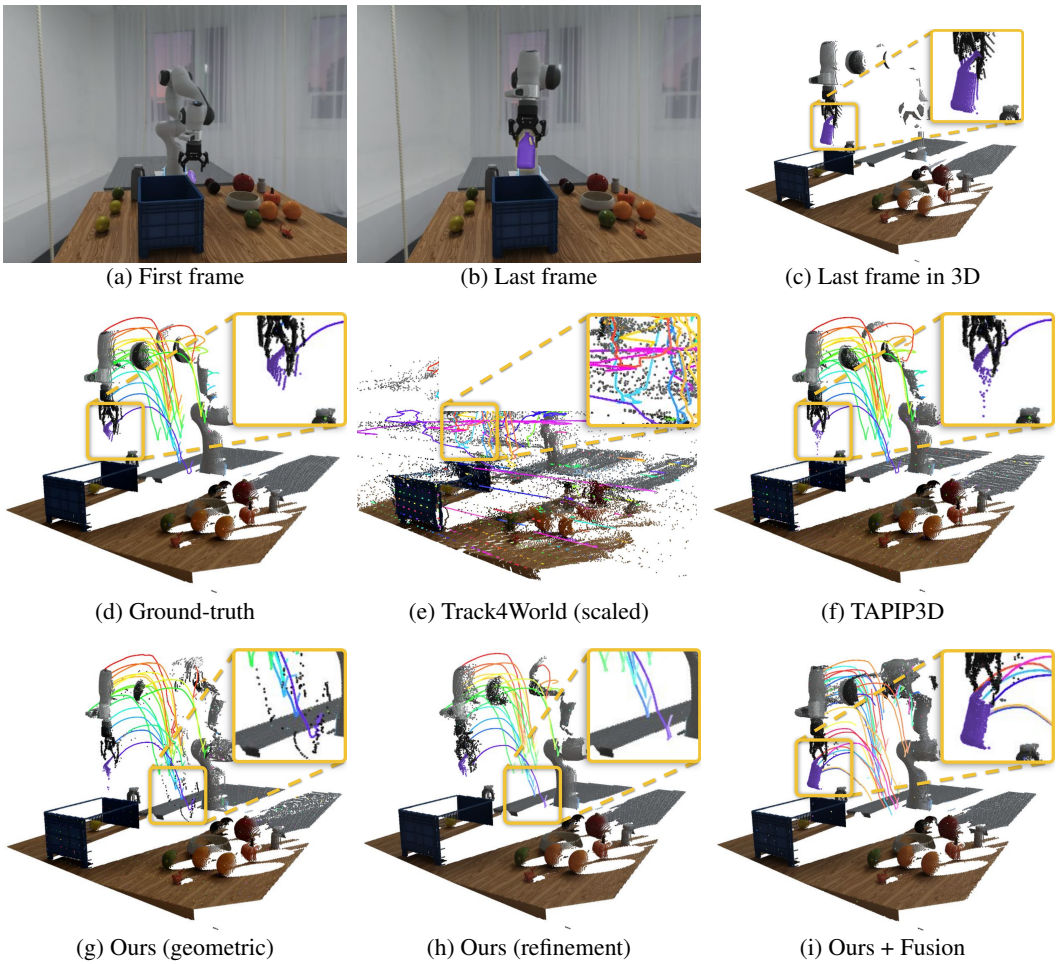
C Training Details

We set $\tau_{\mathrm{motion}} = 2\,\mathrm{mm}$, $k_s = 16$, $k_w = 32$, $L = 3$, $H = 128$, and $n_{\mathrm{layer}} = n_{\mathrm{head}} = 4$. Training uses every episode and camera view from 17 *ArenaPickPlace* scenes for 150k steps. We sample a

random sequence time, initialize the model with ground-truth surface points, roll out one tracking step using ground-truth optical flow for the geometric prior, and minimize the $\mathcal{L}_1$ error between predicted and ground-truth 3D displacement. Optimization uses Adam with batch size 12 on one NVIDIA L40 GPU. The learning rate warms linearly from 3×10^{-6} to 3×10^{-4} over 15k steps and then follows cosine decay to zero; training takes four days. Evaluation replaces ground-truth flow with RAFT or WAFT according to the operating point. The 21 held-out scenes are evaluated using their first episode and camera. Both operating points use the same learned weights and mapping parameters: accurate uses full-resolution RGB-D with WAFT, while fast uses half-resolution RGB-D with RAFT. No fine-tuning is performed on Sim4D or PW-DROID.

D Additional Qualitative Results

Tracking and map maintenance. Fig. SD.1 expands the component comparison in the main paper with sensor context and tracking baselines. The geometric prior leaves hidden arm points behind; refinement propagates their motion, and fusion corrects or removes drifted points and adds the newly revealed bottle. Adding points is excluded from fixed-query EPE for all methods. The baselines omit the bottle because it is absent from their initial queries, not because they cannot track existing queries through occlusion.



(a) First frame (b) Last frame (c) Last frame in 3D

(d) Ground-truth (e) Track4World (scaled) (f) TAPIP3D

(g) Ours (geometric) (h) Ours (refinement) (i) Ours + Fusion

Figure SD.1: **Persistent 3D tracking through occlusion.** (a)–(b) First and last RGB frames of an *ArenaPick-Place* Clutter Plastic test clip; the zoomed region shows the initially occluded bottle. (c) Back-projected frame 97 highlights the disoccluded bottle. (d)–(i) 3D tracks: ground truth, two fixed-query baselines, and our geometric, refined, and full tracking-and-fusion variants. Only the full mapper adds tracks on the newly revealed bottle. Fixed-query EPE (Table SH.3) excludes this adaptive map maintenance.

Hidden-surface reconstruction. Fig. SD.2 adds Black items in bin and Clutter pumpkin to the main paper's Bagels on plate example. The novel views show how much previously observed moving geometry remains represented when hidden from the input camera. ROAMing retains more of this geometry, while the colored points also reveal residual reconstruction errors; quantitative accuracy and completion are reported in Table SH.1.

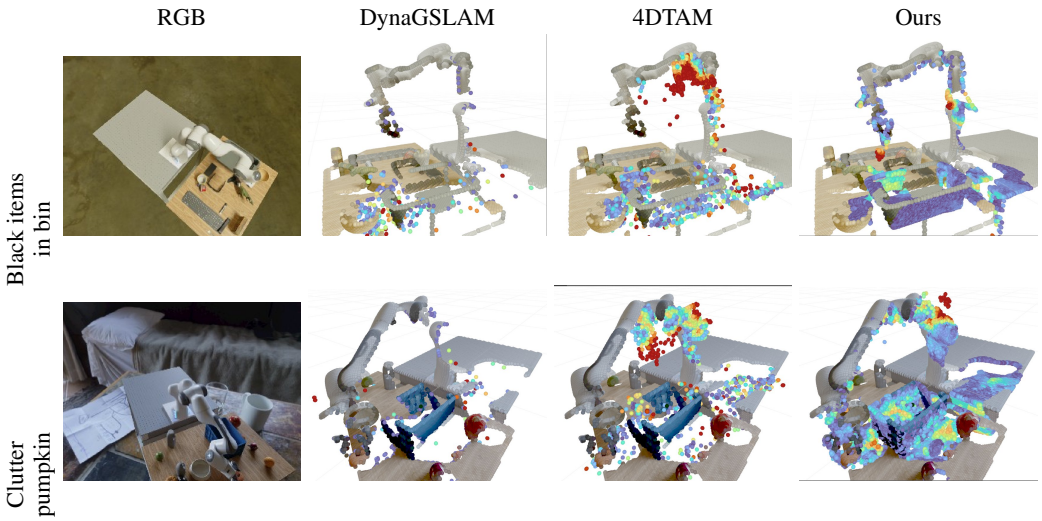


Figure SD.2: **Additional dynamic-hidden reconstruction examples** on *ArenaPickPlace*, complementing Fig. 3. Target points show nearest-surface error using the main figure's 0–5 cm color scale; geometry visible to the input camera provides ground-truth RGB context.

Held-out-view depth. Fig. SD.3 shows Sim4D depth-error maps for Curtain, Mercedes, and Wave toy. These views expose residual errors around moving surfaces, complementing the sequence averages in Table SH.2. The comparison is not a coverage metric: the depth scores exclude pixels without a valid rendered depth (App. F). DynaGSLAM's large errors should be interpreted with its segmentation failure on these sequences in mind (App. E).

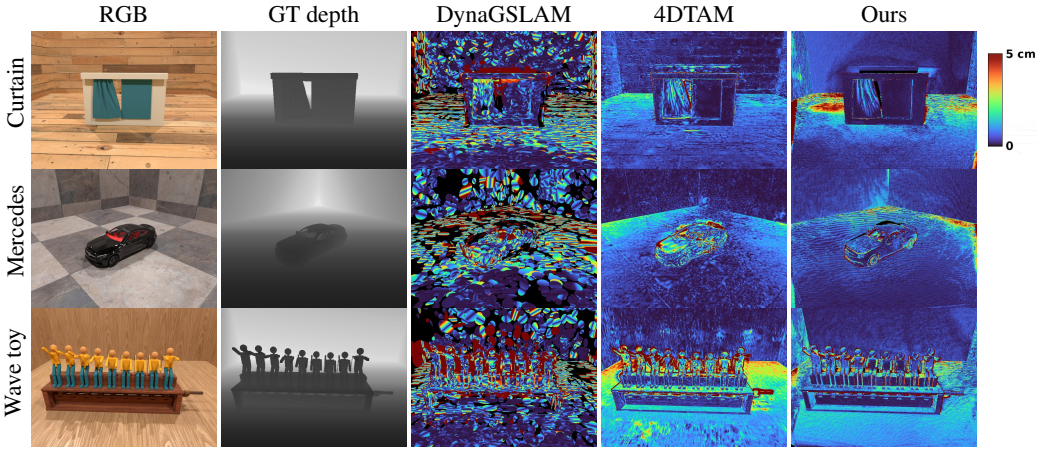


Figure SD.3: **Sim4D qualitative comparison** at held-out viewpoints. After RGB and ground-truth depth, the method columns show absolute depth error on a shared 0–5 cm color scale (blue is lower). DynaGSLAM and ROAMing use poses estimated by 4DTAM.

E Baseline Details

Because 4DTAM cannot process infinite or invalid depth, we insert a background plane for *ArenaPickPlace* and exclude its points from the reconstruction metrics. DynaGSLAM lifts optical flow, as does our geometric prior, and performs motion segmentation with SAM2 [26], seeded from discontinuities in ego-motion-compensated flow. This segmentation requires sufficiently distinct motion and failed on the evaluated sequences. We therefore provide ground-truth masks on *ArenaPickPlace*, where they are available. Sim4D has no ground-truth masks, so DynaGSLAM effectively reduces to a static reconstruction there. DynaGSLAM and our method both require external camera poses; for a fair Sim4D comparison, we use the poses computed by 4DTAM.

F Reconstruction Evaluation Protocol

We adopt the accuracy/completion formulation of NICE-SLAM [27] rather than its full pipeline. Distances are computed between the mapped primitive centers of each method and the ground-truth point cloud. A dynamic scene admits no temporally fused mesh, so the surface-sampling step of the original protocol is unavailable. Completion is consequently sensitive to primitive density, since it counts ground-truth points lying near *some* reconstructed primitive. Accuracy instead measures the fraction of reconstructed centers close to the ground truth and does not penalize missing surfaces directly. Table SH.1 therefore reports the mean primitive count per frame alongside both metrics. Points seen earlier and occluded at evaluation time are retained and scored in full; only geometry that no input view ever saw is excluded, since a map built purely from observed evidence cannot represent it.

Sim4D does not provide scene-space ground-truth point clouds. At each benchmark frame, we render the reconstructed primitives from the held-out evaluation camera and compute absolute depth error over the intersection of valid rendered and ground-truth depth pixels. Pixels where no method primitive renders are excluded, so this metric measures rendered-depth accuracy rather than completion. We average over frames to obtain each sequence score and then average sequence scores. DynaGSLAM and ROAMing use the camera poses estimated by 4DTAM.

G Dataset and Annotation Details

The dataset contains 38 ray-traced tabletop scenes, each with 20 robot-arm trajectories captured by 10 hemispherically placed cameras for 200 frames. Outputs include RGB, depth, optical flow, per-frame object meshes and poses, and surface trajectories. We construct the evaluation ground truth from Isaac Lab Arena outputs and assign scene-state labels as follows.

G.1 Surface sampling

Each object in a scene is provided as a triangle mesh in its own object-local frame, together with a per-frame world pose track. We draw points uniformly over mesh surface area, storing a position and a surface normal per sample, and keep them in the object-local frame. At evaluation time the samples of each object are rigidly warped into the world frame by that object's pose at the current frame. The ground-truth is therefore re-posed rather than re-sampled, and a given point retains its identity across the sequence. Sampling draws from a dedicated seeded generator, which gives every method a bit-identical point set and attributes any metric difference to the reconstruction alone.

G.2 Point budget

We sample 10^6 points per scene, allocated across objects in proportion to mesh surface area. This makes sample density uniform over the scene rather than uniform per object. No scene-state category information enters the allocation, since the static and dynamic labels are assigned afterwards. The division of the budget therefore follows scene geometry: static structure such as the table, bins,

walls and fixed props receives roughly 94% of the points. Averaged over the test scenes, dynamic objects receive $\sim 5.7 \times 10^4$ points, of which the articulated robot accounts for $\sim 4.8 \times 10^4$ and the manipulated objects for the remainder.

Accuracy and completion are measured against sets of different density. Accuracy (reconstruction $\rightarrow$ ground-truth) uses the full 10^6-point set. Each reconstructed point is matched to its nearest ground-truth point and inherits that point's category and visibility labels. A sparse ground-truth would both inflate nearest-neighbor distances and mislabel reconstructed points near a category boundary. Completion (ground-truth $\rightarrow$ reconstruction) uses a fixed, evenly strided 10^5-point subset, a factor of ten smaller. This estimate is the fraction of ground-truth points within a distance threshold of the reconstruction. It converges with fewer ground-truth samples, although it remains sensitive to reconstruction primitive density. The reduction keeps the per-frame nearest-neighbor search tractable and reduces benchmarking time.

G.3 Scene-state annotations

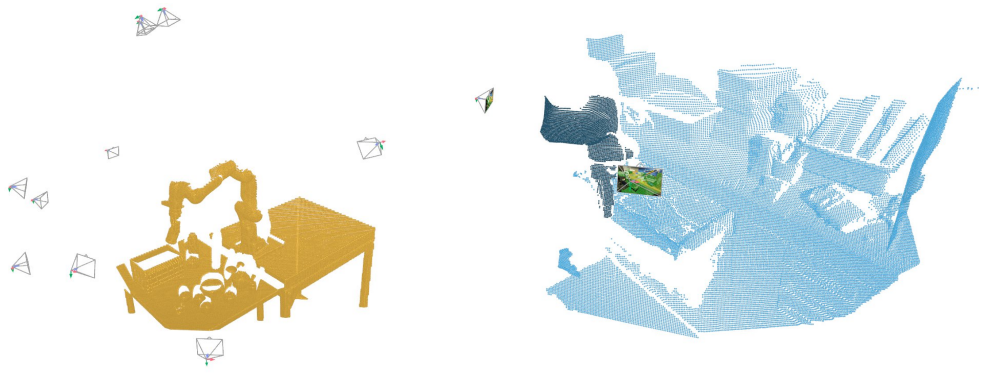
Every ground-truth point carries two independent labels.

Static or dynamic is a per-object property derived from the pose track alone. An object is dynamic if its pose at any scored frame departs from its pose at the first scored frame by more than $1\,\mathrm{cm}$ in translation or 5° in rotation, and static otherwise. The label is constant over the sequence: an object that is moved once counts as dynamic throughout, including the frames before and after it moves.

Visible or hidden is a per-point, per-frame property. A point is visible at a frame if it projects inside the image, its normal faces the camera, and its depth agrees with the ground-truth depth map to within $3\,\mathrm{cm}$. Back-facing, occluded, and out-of-view points are hidden. Points that are never visible at any frame, such as interior surfaces, object undersides, and geometry outside the camera frustum throughout the sequence, are excluded from evaluation because they have no observed evidence. Temporarily out-of-view points remain eligible if observed in another frame. All reported buckets are intersections of these two labels over the ever-observed subset.

G.4 Visualization

Tracking queries. Fig. SG.1 shows the first-frame, pixel-aligned 3D query points used for tracking. *ArenaPickPlace* provides synthetic ground-truth trajectories, whereas PW-DROID provides real-world pseudo-ground-truth trajectories. Restoring PW-DROID's omitted robot-arm points makes it possible to score the robot's visible and hidden subsets; the annotation and tracking protocol is detailed in App. H.



ArenaPickPlace PW-DROID

Figure SG.1: **First-frame 3D tracking queries.** Per-view, pixel-aligned reference points in *ArenaPickPlace* (synthetic ground truth) and PW-DROID (real-world pseudo-ground-truth trajectories). PW-DROID includes restored robot-arm points absent from its original annotations.

Synthetic annotations. Fig. SG.2 illustrates the RGB, depth, optical flow, and scene-state annotations of *ArenaPickPlace*. Ray-traced RGB uses HDR backgrounds and physically based lighting; depth and flow come directly from simulation. Novel-view renderings expose the visible/hidden and static/dynamic subsets, including surfaces hidden in the input view.

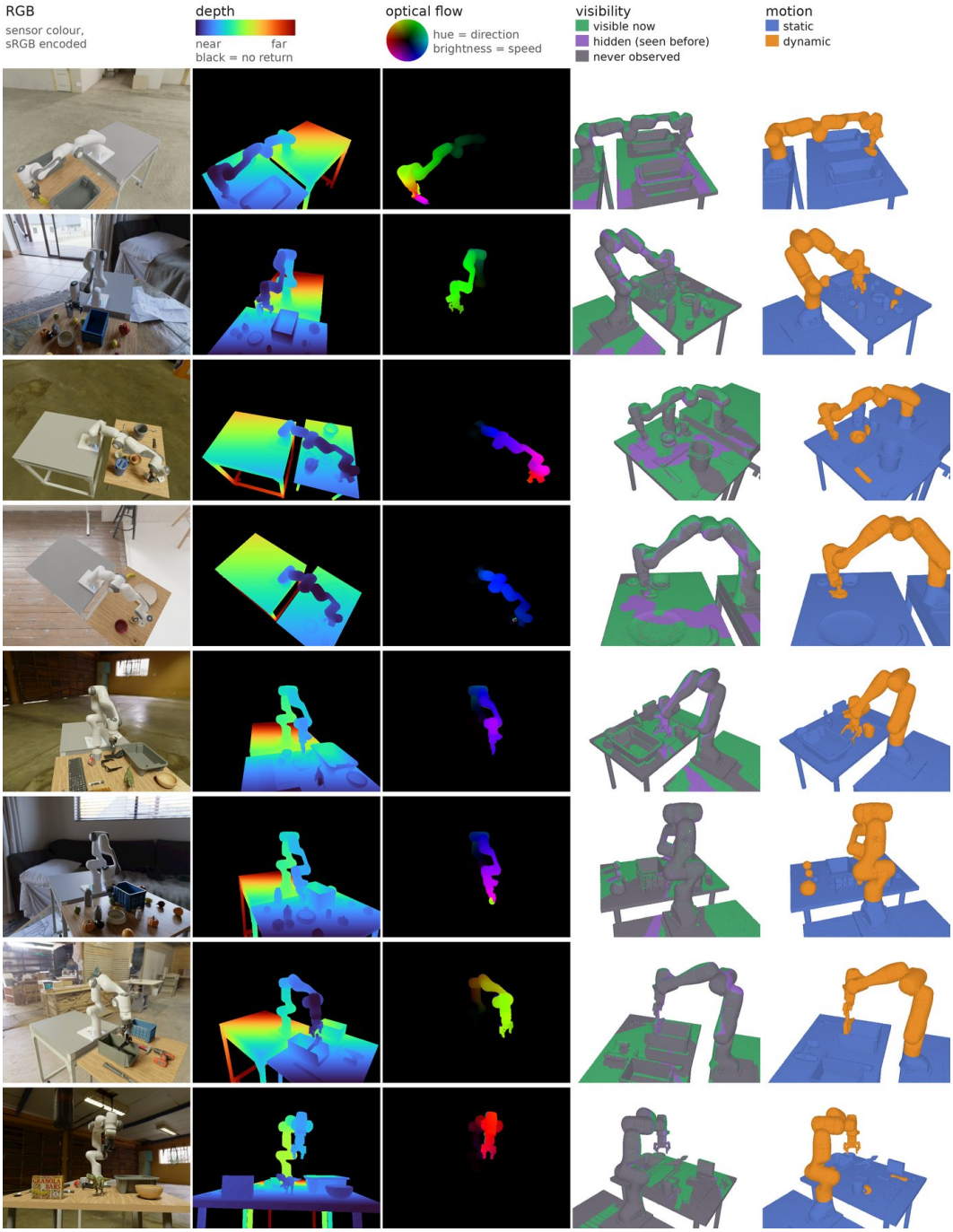


Figure SG.2: ***ArenaPickPlace* sequences and annotations.** Columns show RGB, depth, optical flow, visibility, and motion labels. The final two columns use novel views to reveal hidden surfaces and distinguish the robot from other moving objects.

H Additional Quantitative Results

H.1 Full reconstruction results

Table SH.1 restates the main paper's reconstruction results for all four motion/visibility subsets. Accurate ROAMing reaches 90.2% dynamic-hidden completion, versus 60.7% for DynaGSLAM and 55.7% for 4DTAM. Fast ROAMing retains 88.3% completion with 31k primitives rather than 102k, although its dynamic-hidden accuracy is slightly below DynaGSLAM's (86.0% versus 86.9%).

Table SH.1: Full reconstruction results on *ArenaPickPlace*. Each cell reports accuracy/completion at 3 cm (%, ↑); the Total column also reports the mean primitive count per frame. Total and category metrics are aggregated separately. DynaGSLAM uses ground-truth object masks.

	Total	**Static**		**Dynamic**	
Method		**Visible**	**Hidden**	**Visible**	**Hidden**
DynaGSLAM	96.4/60.7 30k pts	83.1/56.8 36.7%	80.7/57.5 1.1%	98.8/99.9 60.0%	86.9/60.7 1.0%
4DTAM	81.4/82.9 12k pts	89.6/86.6 66.7%	75.6/77.9 2.6%	74.7/66.2 16.7%	56.1/55.7 16.7%
Ours (accurate)	**97.4/99.1** 102k pts	97.5/**99.6** 75.5%	**95.5/98.9** 5.2%	**99.4/100.0** 15.7%	**89.7/90.2** 3.3%
Ours (fast)	96.9/98.7 31k pts	**97.6**/99.3 64.5%	**95.5/98.9** 4.8%	98.1/**100.0** 23.9%	86.0/88.3 6.1%

The dynamic-hidden primitive shares are 1.0% for DynaGSLAM, 16.7% for 4DTAM, 3.3% for Ours (accurate), and 6.1% for Ours (fast). Thus, similar total map sizes do not imply matched category density; the completion comparison is not controlled for this factor.

Table SH.2 reports all eight Sim4D sequences. Accurate ROAMing and 4DTAM have similar mean depth error (1.77 and 1.79 cm), but performance varies by sequence: ROAMing has lower error on five sequences, while 4DTAM leads on Curtain, Shoe rack, and Wave toy. The fast configuration averages 2.08 cm. These overlap-only depth scores do not establish equal surface coverage across methods.

Table SH.2: Full Sim4D held-out-view, overlap-only depth L1 error in cm (↓). All methods marked † use poses computed by 4DTAM.

	DynaGSLAM†	**4DTAM**	**Ours (accurate)**†	**Ours (fast)**†
Curtain	5.81	**0.83**	1.06	1.09
Flag	5.97	3.18	**2.85**	2.97
Mercedes	3.17	0.86	**0.69**	0.82
Mod. veh.	9.37	1.37	**0.87**	1.14
Rhino	7.78	1.17	**0.98**	1.36
Shoe rack	5.13	**1.28**	1.48	1.62
Water drop	6.41	3.82	**2.77**	3.42
Wave toy	17.12	**1.82**	3.44	4.26
Mean	7.59	1.79	**1.77**	2.08

H.2 Full tracking and runtime results

For fixed-query EPE, every method receives the same pixel-aligned first-frame 3D points and tracks them through the full sequence. ROAMing evaluates its motion module on this identity-preserving set; fusion-based spawning and pruning are disabled for the metric, so no query is replaced. PointWorld's PW-DROID augments DROID [28] with 3D real-world pseudo-ground-truth trajectories for 12-frame clips; PW-DROID contains approximately 2% occluded points, versus 7% in *ArenaPick-Place*. We restore its omitted robot-arm annotations by sampling first-frame pixel-aligned points on the provided link meshes and transforming them with the per-frame link poses. A robot point is

visible when it projects inside the image and agrees with the provided depth within 3 cm; otherwise it is hidden. TAPIP3D uses 16-frame windows with stride 8, whereas Track4World processes the full video. ROAMing updates from the current frame pair without future observations. Track4World exhibits substantial scale drift, so for each sequence its reported "scaled" variant uses

$$s^\star = \arg\min_s \sum_{(i,t)\in\mathcal{V}} \|s\,\widehat{\mathbf{p}}_t^i - \mathbf{p}_t^i\|_2^2,$$

where $\mathcal{V}$ contains all query–frame pairs with valid visible reference trajectories. The same $s^\star$ scales every trajectory in that sequence; this alignment uses evaluation ground truth and is not causal.

Table SH.3 reports the full EPE breakdown. TAPIP3D has the lowest EPE in every reported subset; ROAMing trades accuracy for causal updates. Hidden moving points remain challenging: accurate ROAMing's *ArenaPickPlace* dynamic-hidden EPE is 0.0751 m, compared with 0.0329 m for dynamic-visible points. The supplementary tables retain the original results in meters; the main paper expresses EPE in centimeters (e.g., 0.0099 m is 0.99 cm).

Table SH.3: Full fixed-query EPE3D tracking comparison in meters (↓), excluding fusion-based spawning and pruning. Track4World (scaled) uses sequence-level ground-truth scale alignment.

ArenaPickPlace

Method	All	Static vis.	Static hidden	Dynamic vis.	Dynamic hidden
Track4World (scaled)	0.1361	0.1183	0.2013	0.1800	0.2959
TAPIP3D	**0.0053**	**0.0026**	**0.0055**	**0.0112**	**0.0426**
Ours (fast)	0.0156	0.0035	0.0224	0.0583	0.1042
Ours (accurate)	0.0099	0.0027	0.0211	0.0329	0.0751

PW-DROID

Method	All	Scene	Robot vis.	Robot hidden
Track4World (scaled)	0.1221	0.1225	0.1340	0.1556
TAPIP3D	**0.0035**	**0.0033**	**0.0039**	**0.0096**
Ours (fast)	0.0072	0.0047	0.0152	0.0309
Ours (accurate)	0.0054	0.0042	0.0078	0.0185

Table SH.4 complements the tracking errors with runtime and peak GPU memory. Fast ROAMing takes 56/58 ms per frame pair on *ArenaPickPlace*/PW-DROID, using 3.1/1.9 GiB, and includes both tracking and fusion. Input priors are precomputed. Whole-clip averages are not causal per-frame latency, and the L40 measurements for memory-intensive baselines should not be read as same-hardware speedups over RTX 5090 results.

Table SH.4: Backend runtime and peak GPU memory with input priors precomputed; these are not end-to-end timings. Whole-clip runtimes are amortized over each sequence. Ours includes tracking and fusion. Non-† results use an NVIDIA GeForce RTX 5090 and Intel Core i9-14900K; † marks NVIDIA L40 results for methods exceeding the RTX 5090's 31.4 GiB.

	ArenaPickPlace		**PW-DROID**	
Method	**runtime (ms)**	**GiB**	**runtime (ms)**	**GiB**
Track4World	661†	35.6†	146	26.3
TAPIP3D	11,594†	42.6†	2,924	11.1
4DTAM	228	8.6	—	—
DynaGSLAM	149	**1.1**	—	—
Ours (accurate)	186	4.8	254	10.4
Ours (fast)	**56**	3.1	**58**	**1.9**

H.3 Tracking metrics and ablations

We report the standard 3D tracking metrics present in Track4World and the ones used in TAPIP3D, introduced by TAPVid-3D [29]. EPE3D/EPE2D measure 3D/2D tracking error in meters/pixels;

Abs Rel is relative 3D error, AccS/AccR and their 2D variants measure strict/relaxed accuracy, APD_{3D} and AJ_{3D} measure position and visibility-aware tracking accuracy, while OA measures occlusion accuracy. The $\delta < 1.25$ column reports ratio-threshold accuracy. All metrics other than EPE3D/EPE2D are dimensionless, with accuracy scores expressed as fractions rather than percentages. "Occluded" in the detailed tables denotes the non-visible subset called "hidden" in the main paper and EPE summary. Dashes preserve unreported metrics; they are not zero scores.

The RAFT ablation changes the flow estimator at the accurate configuration's full resolution. In contrast, fast changes both flow estimator and input resolution, so it is an operating-point comparison, not an isolated flow ablation. Pixel-valued errors also depend on image resolution; EPE3D is the common metric used to compare the operating points in the main paper.

***ArenaPickPlace* ablations.** Table SH.5 shows that removing the 2 mm motion threshold increases aggregate EPE3D from 0.0099 to 0.0221 m. Replacing WAFT with RAFT at full resolution gives 0.0122 m. These comparisons support the threshold's role in limiting accumulated drift and quantify sensitivity to the flow prior; they do not isolate the learned refiner, whose contribution is illustrated qualitatively in Fig. SD.1.

PW-DROID ablations. Table SH.6 shows a much smaller aggregate threshold effect on the 12-frame real-world clips: EPE3D is 0.0055 m without the threshold and 0.0054 m with it. The full-resolution RAFT variant reaches 0.0063 m. Thus, the pronounced aggregate threshold benefit on the longer synthetic sequences should not be claimed uniformly across datasets.

Table SH.5: Aggregate tracking ablations on *ArenaPickPlace*, evaluated over all query points. EPE3D is in meters, EPE2D in pixels, and other scores are dimensionless (App. H.3).

Method	Abs Rel↓	$\delta < 1.25 \uparrow$	EPE3D↓	AccS↑	AccR↑	EPE2D↓	$\text{AccS}_{2D} \uparrow$	$\text{AccR}_{2D} \uparrow$	$\text{AJ}_{3D} \uparrow$	$\text{APD}_{3D} \uparrow$	OA↑
Ours (accurate) w/o τ_{motion}	0.0164	0.9988	0.0221	0.9070	0.9811	5.5263	0.2026	0.5365	0.5369	0.6631	0.9199
Ours (accurate) w/ RAFT	0.0089	0.9987	0.0122	0.9182	0.9724	3.1696	0.7525	0.8062	0.8001	0.8870	0.9528
Ours (accurate)	0.0072	0.9989	0.0099	0.9390	0.9849	2.5396	0.7555	0.8180	0.8162	0.8976	0.9562
Ours (fast)	0.0116	0.9982	0.0156	0.8874	0.9519	2.2187	0.7748	0.8310	0.7699	0.8695	0.9539

Table SH.6: Aggregate tracking ablations on PW-DROID, evaluated over all query points. EPE3D is in meters, EPE2D in pixels, and other scores are dimensionless (App. H.3).

Method	Abs Rel↓	$\delta < 1.25 \uparrow$	EPE3D↓	AccS↑	AccR↑	EPE2D↓	$\text{AccS}_{2D} \uparrow$	$\text{AccR}_{2D} \uparrow$	$\text{AJ}_{3D} \uparrow$	$\text{APD}_{3D} \uparrow$	OA↑
Ours (accurate) w/o τ_{motion}	0.0084	0.9981	0.0055	0.9888	0.9972	2.0395	0.8357	0.9477	0.7548	0.8568	0.9459
Ours (accurate) w/ RAFT	0.0095	0.9980	0.0063	0.9854	0.9966	2.2351	0.7891	0.9167	0.7300	0.8396	0.9393
Ours (accurate)	0.0083	0.9981	0.0054	0.9887	0.9972	2.0109	0.8362	0.9472	0.7548	0.8582	0.9437
Ours (fast)	0.0111	0.9983	0.0072	0.9764	0.9950	1.1918	0.8326	0.9394	0.6992	0.8301	0.9111

Motion and visibility breakdown. Table SH.7 shows the full *ArenaPickPlace* metric suite, including unscaled and scaled Track4World. Scale alignment reduces Track4World's 3D error but leaves a substantial gap to the RGB-D methods. For accurate ROAMing, dynamic-hidden EPE3D is 0.0751 m versus 0.0211 m for static-hidden points, exposing the difficulty of propagating hidden motion despite low aggregate error.

Table SH.7: Quantitative comparison on 3D tracking metrics on *ArenaPickPlace*, separated by motion class and visibility. Static includes world and static-object points; Dynamic includes rigid and articulated moving-object points. ROAMing uses its stated accurate/fast input resolutions. EPE3D is in meters and EPE2D in pixels; other scores are dimensionless.

Method	**Motion**	**Vis.**	Abs Rel↓	$\delta < 1.25$ ↑	EPE3D↓	AccS↑	AccR↑	EPE2D↓	AccS_{2D} ↑	AccR_{2D} ↑	AJ_{3D} ↑	APD_{3D} ↑	OA↑
Track4World [6]	Static	Visible	0.3975	0.0749	0.6481	0.0059	0.0252	12.1933	0.0020	0.0458	0.0017	0.0032	0.9921
		Occluded	0.3668	0.1532	0.6186	0.0137	0.0466	11.1006	0.0097	0.0709	—	—	0.7581
	Dynamic	Visible	0.4041	0.1235	0.5154	0.0126	0.0341	10.9380	0.0218	0.1555	0.0042	0.0078	0.9416
		Occluded	0.4097	0.1722	0.5843	0.0105	0.0484	20.4121	0.0134	0.0932	—	—	0.5598
Track4World (scaled) [6]	Static	Visible	0.0732	0.9676	0.1183	0.3717	0.6723	12.1963	0.0020	0.0458	0.1568	0.2142	0.9921
		Occluded	0.1166	0.8936	0.2013	0.2014	0.4704	11.1006	0.0097	0.0709	—	—	0.7581
	Dynamic	Visible	0.1449	0.8690	0.1800	0.2575	0.4980	10.9383	0.0218	0.1555	0.0892	0.1422	0.9416
		Occluded	0.2094	0.7714	0.2959	0.1352	0.3258	20.4121	0.0134	0.0932	—	—	0.5598
TAPIP3D [7]	Static	Visible	0.0016	1.0000	0.0026	0.9995	0.9997	0.6186	0.8222	0.9765	0.9054	0.9777	0.9410
		Occluded	0.0033	0.9999	0.0055	0.9958	0.9979	1.2937	0.6130	0.9190	—	—	0.9663
	Dynamic	Visible	0.0087	1.0000	0.0112	0.9741	0.9960	3.2786	0.3456	0.6609	0.6784	0.7913	0.9506
		Occluded	0.0284	0.9817	0.0426	0.7875	0.9577	9.5731	0.1359	0.3486	—	—	0.6269
Ours (accurate)	Static	Visible	0.0017	1.0000	0.0027	0.9884	0.9964	0.5804	0.9070	0.9443	0.9317	0.9620	0.9852
		Occluded	0.0131	1.0000	0.0211	0.8797	0.9834	4.5434	0.3114	0.5409	—	—	0.9025
	Dynamic	Visible	0.0259	0.9984	0.0329	0.7696	0.9496	9.2784	0.1903	0.3420	0.4120	0.5408	0.8636
		Occluded	0.0517	0.9722	0.0751	0.5185	0.8517	18.6915	0.0254	0.1141	—	—	0.6430
Ours (fast)	Static	Visible	0.0022	1.0000	0.0035	0.9783	0.9939	0.4384	0.9230	0.9592	0.9241	0.9550	0.9836
		Occluded	0.0141	1.0000	0.0224	0.8319	0.9593	2.7964	0.5597	0.7112	—	—	0.8935
	Dynamic	Visible	0.0463	0.9961	0.0583	0.5513	0.7998	8.9443	0.2019	0.3332	0.2775	0.3965	0.9007
		Occluded	0.0731	0.9646	0.1042	0.3317	0.6429	13.6447	0.0701	0.1931	—	—	0.4828

Real-world scene and robot breakdown. Table SH.8 separates PW-DROID scene and restored robot annotations. Accurate ROAMing's EPE3D is 0.0042 m on scene points, 0.0078 m on visible robot points, and 0.0185 m on hidden robot points; TAPIP3D leads in all three subsets. Short-clip tracking does not establish persistent real-world reconstruction.

Table SH.8: PW-DROID tracking by scene/robot and visibility. All/Occluded equals Robot/Occluded; Scene/All equals Scene/Visible (there are no occluded scene queries). EPE3D is in meters, EPE2D in pixels; other scores are dimensionless.

Method	**Points**	**Vis.**	Abs Rel↓	$\delta < 1.25$ ↑	EPE3D↓	AccS↑	AccR↑	EPE2D↓	AccS_{2D} ↑	AccR_{2D} ↑	AJ_{3D} ↑	APD_{3D} ↑	OA↑
Track4World [6]	All	All	0.2945	0.6014	0.2004	0.0984	0.3340	14.1253	0.0078	0.0671	0.0187	0.0312	0.9794
		Visible	0.2908	0.6073	0.1996	0.0986	0.3355	12.7926	0.0079	0.0678	0.0188	0.0312	0.9943
	Scene	Visible	0.2535	0.6821	0.1961	0.0907	0.3410	12.3098	0.0082	0.0704	0.0211	0.0339	0.9972
	Robot	All	0.4164	0.4108	0.2217	0.0800	0.2791	17.4852	0.0040	0.0340	0.0071	0.0126	0.9584
		Visible	0.4102	0.4169	0.2190	0.0829	0.2855	14.2900	0.0040	0.0350	0.0072	0.0126	0.9815
		Occluded	0.4526	0.4122	0.2535	0.0684	0.2560	20.5188	0.0056	0.0453	—	—	0.4623
Track4World (scaled) [6]	All	All	0.1642	0.8223	0.1221	0.2591	0.6452	14.1253	0.0078	0.0671	0.0433	0.0712	0.9794
		Visible	0.1619	0.8262	0.1218	0.2610	0.6469	12.7926	0.0079	0.0678	0.0437	0.0712	0.9943
	Scene	Visible	0.1431	0.8680	0.1225	0.2511	0.6514	12.3098	0.0082	0.0704	0.0516	0.0813	0.9972
	Robot	All	0.2410	0.6712	0.1355	0.2015	0.5524	17.4852	0.0040	0.0340	0.0150	0.0264	0.9584
		Visible	0.2373	0.6777	0.1340	0.2092	0.5590	14.2900	0.0040	0.0350	0.0152	0.0264	0.9815
		Occluded	0.2655	0.6599	0.1556	0.1610	0.5101	20.5188	0.0056	0.0453	—	—	0.4623
TAPIP3D [7]	All	All	0.0050	0.9995	0.0035	0.9954	0.9990	1.6631	0.8832	0.9679	0.8240	0.9150	0.9312
		Visible	0.0045	0.9998	0.0033	0.9962	0.9992	0.4819	0.8975	0.9769	0.8300	0.9150	0.9354
	Scene	Visible	0.0039	0.9998	0.0033	0.9948	0.9989	0.3310	0.9507	0.9880	0.8815	0.9386	0.9654
	Robot	All	0.0089	0.9986	0.0046	0.9947	0.9987	3.7674	0.6819	0.9090	0.6564	0.8277	0.8385
		Visible	0.0074	0.9995	0.0039	0.9977	0.9994	1.0466	0.7157	0.9361	0.6623	0.8277	0.8289
		Occluded	0.0170	0.9976	0.0096	0.9816	0.9961	6.0044	0.4183	0.7731	—	—	0.7743
Ours (accurate)	All	All	0.0083	0.9981	0.0054	0.9887	0.9972	2.0109	0.8362	0.9472	0.7548	0.8582	0.9437
		Visible	0.0071	0.9992	0.0049	0.9917	0.9981	0.6498	0.8515	0.9592	0.7650	0.8582	0.9568
	Scene	Visible	0.0050	0.9997	0.0042	0.9906	0.9980	0.3921	0.9278	0.9774	0.8460	0.9122	0.9661
	Robot	All	0.0177	0.9949	0.0093	0.9812	0.9949	4.6578	0.5655	0.8616	0.5395	0.6773	0.9011
		Visible	0.0144	0.9982	0.0078	0.9922	0.9979	1.4277	0.6009	0.8996	0.5520	0.6773	0.9173
		Occluded	0.0328	0.9908	0.0185	0.9359	0.9892	7.9763	0.2513	0.6439	—	—	0.6375
Ours (fast)	All	All	0.0111	0.9983	0.0072	0.9764	0.9950	1.1918	0.8326	0.9394	0.6992	0.8301	0.9111
		Visible	0.0098	0.9991	0.0067	0.9803	0.9960	0.5594	0.8478	0.9519	0.7092	0.8301	0.9242
	Scene	Visible	0.0057	0.9996	0.0047	0.9858	0.9972	0.2539	0.9482	0.9821	0.8148	0.9068	0.9376
	Robot	All	0.0293	0.9936	0.0168	0.9289	0.9832	3.1911	0.4783	0.7949	0.4146	0.5644	0.8445
		Visible	0.0258	0.9962	0.0152	0.9423	0.9866	1.6884	0.5090	0.8320	0.4267	0.5644	0.8654
		Occluded	0.0501	0.9890	0.0309	0.8277	0.9597	5.4567	0.1880	0.5597	—	—	0.5665

I Additional Method Details

I.1 Surfel Fusion and Map Update

We maintain a dense surface reconstruction as an unstructured set of surfels, updated incrementally from posed RGB-D frames. This section describes the static mapping core: per-frame integration (Sec. I.1.1) and the confidence-based lifecycle that removes stale geometry (Sec. I.1.7). The dynamic motion module (Sec. 2.1) propagates the map before this standard fusion pipeline runs.

I.1.1 Integration overview

An input frame is $\mathcal{F}_t = (D_t, I_t, K, \mathbf{T}_{W \leftarrow C_t})$. Integrating $\mathcal{F}_t$ into the propagated map $\widetilde{\mathcal{M}}_t$ proceeds in four steps: (1) estimate a normal image from the depth image, (2) associate map surfels with measurements by projecting them into the current view, (3) fuse the associated pairs with an information filter, and (4) rasterise the map into the current view and spawn new surfels only where it leaves the observed surface uncovered.

I.1.2 Normal estimation and measurement validity

We back-project the depth image into an organised vertex map $V(\mathbf{u}) = \pi^{-1}(\mathbf{u}, D_t(\mathbf{u}))$ and obtain normals from central differences on the pixel grid,

$$\mathbf{n}_C(\mathbf{u}) \;=\; \frac{\partial_u V \times \partial_v V}{\|\partial_u V \times \partial_v V\|}, \tag{4}$$

oriented so that $\mathbf{n}_C$ points away from the camera. A pixel carries a valid normal only if it and its four neighbours have valid depth, the cross product is non-degenerate, and the incidence satisfies $|[n_C]_z| > \cos\theta_{\max}$. Pixels outside the integration range $[z_{\min}, z_{\max}]$ are masked out.

I.1.3 Surfel initialisation

A surfel created at pixel $\mathbf{u}$ with depth $z = D_t(\mathbf{u})$ takes its center from the back-projected point and its normal from Eq. (4), both mapped to the world frame by $\mathbf{T}_{W \leftarrow C_t}$. Color is taken from $I_t(\mathbf{u})$. Following Weise et al. [30], the radius is set to

$$r \;=\; \mathrm{clip}\!\left(\frac{z}{\sqrt{2}\, f\, [n_C]_z},\; r_{\min},\; r_{\max}\right), \qquad [r_{\min}, r_{\max}] = [1, 20]\,\mathrm{cm}. \tag{5}$$

Here, f is the camera focal length. The term $z/\sqrt{2}f$ is the half-diagonal of the pixel footprint back-projected onto a fronto-parallel surface at depth z, so neighbouring disks overlap and rasterise to a hole-free image (Sec. I.1.6). The factor $1/[n_C]_z$ enlarges the disk on tilted surfaces, whose footprint is stretched along the direction of steepest depth change. Clipping prevents degenerate radii.

We use a constant-relative-error heuristic and initialise the estimation variances as

$$\sigma_p^2 = \alpha_p^2 z^2, \qquad \sigma_n^2 = \alpha_n^2 z^2, \tag{6}$$

with $\alpha_p = 2 \cdot 10^{-2}$ and $\alpha_n = 10^{-2}$. Distant measurements therefore enter the fusion of Sec. I.1.5 with proportionally less weight.

I.1.4 Data association

Data association is projective. Every map surfel is transformed into the current camera frame, $\mathbf{p}_C = \mathbf{T}_{W \leftarrow C_t}^{-1} \mathbf{p}_W$, and projected to $\mathbf{u}_i = \pi(\mathbf{p}_C^i)$ with projected depth $z_i = [\mathbf{p}_C^i]_z$. For surfels landing inside the image we bilinearly sample depth, color and normal at $\mathbf{u}_i$ and assemble a *measurement surfel* from those samples using Eqs. (5)–(6), giving a one-to-one correspondence with the map. In particular, the depth measurement for surfel S^i is

$$\begin{aligned} d_t^i &= D_t[\mathbf{u}^i] \\ &= D_t[\pi(\mathbf{T}_{W \leftarrow C_t}^{-1} \mathbf{p}_W^i)] \end{aligned} \tag{7}$$

where $\mathbf{p}_W^i$ is the center of the ith surfel in the world frame and $D[\cdot]$ denotes bilinear sampling of the depth image, and the corresponding position measurement is

$$\mathbf{q}_t^i = \mathbf{T}_{W \leftarrow C_t} \pi^{-1}(\mathbf{u}^i, d_t^i). \tag{8}$$

The association is accepted when the sampled measurement is valid, is not excluded by the masks above, and is depth-consistent with the surfel,

$$\left| z_i - D_t(\mathbf{u}_i) \right| < \tau, \qquad \tau = 5\,\mathrm{cm}. \tag{9}$$

The threshold rejects correspondences across occlusion boundaries: a surfel whose center projects onto a surface at a different range is not fused with it. The signed discrepancy then distinguishes free-space contradictions, which reduce confidence, from occluded geometry, which is retained (Sec. I.1.7).

I.1.5 Fusion

Accepted pairs are fused in place with an information filter, i.e. a Kalman filter written in the information form, in the manner of EGG-Fusion [16]. Under the isotropic assumption the information matrix of a state collapses to the scalar $\omega \triangleq \sigma^{-2}$, and the update of a prior $(\mathbf{x}^-, \omega^-)$ with a measurement $(\mathbf{x}^z, \omega^z)$ is a pair of additions,

$$\omega^+ = \omega^- + \omega^z, \qquad \mathbf{x}^+ = \frac{\omega^- \mathbf{x}^- + \omega^z \mathbf{x}^z}{\omega^+}. \tag{10}$$

We apply Eq. (10) independently to the center (with $\omega = 1/\sigma_p^2$) and to the normal (with $\omega = 1/\sigma_n^2$), renormalising the fused normal to unit length afterwards. Repeated observation therefore drives σ_p^2 and σ_n^2 towards zero, and the surfel becomes progressively insensitive to new measurements, which is the intended behaviour for static geometry. The learned propagation stage updates centers only. It leaves a hidden surfel's normal unchanged; the update above refreshes the normal once the surface is observed again.

Color and radius carry no dedicated uncertainty and are fused with the positional weights of Eq. (10). The radius is additionally constrained to be non-increasing,

$$r^+ \;=\; \mathrm{clip}\big(\min(r^-, \tilde{r}),\; r_{\min},\; r_{\max}\big), \tag{11}$$

where $\tilde{r}$ is the fused radius. A close-up observation of a surface refines it to a finer scale permanently, and a later distant or grazing view of the same surface, which by Eq. (5) would propose a large disk, cannot coarsen the reconstruction again.

I.1.6 Surfel Generation

New geometry is added only where the current map fails to explain the observed surface. We rasterise the map into the current view: a surfel disk of radius r at depth z projects to an ellipse centered at $\pi(\mathbf{p}_C)$ with semi-major axis $a = rf/z$ and semi-minor axis

$$b \;=\; a\,\big(\mathbf{n}_C^\top \mathbf{d}\big)_+, \qquad \mathbf{d} = \mathbf{p}_C / \|\mathbf{p}_C\|, \tag{12}$$

foreshortened by the cosine between the normal and the viewing ray $\mathbf{d}$, and oriented such that the minor axis is aligned with the image-plane projection of the normal. Back-facing and edge-on surfels have $b = 0$ and are skipped.

A pixel is marked *covered* if it falls inside such an ellipse *and* the projected depth of that surfel agrees with the measurement there in the sense of Eq. (9). Without this depth gate a surfel lying far behind a newly observed, closer surface would claim the pixel and suppress the reconstruction of that surface; with it, occluded map geometry no longer blocks spawning. New surfels are then initialised at every valid, uncovered pixel by the procedure of Sec. I.1.3.

Rasterisation is implemented as a GPU kernel that visits, per surfel, a square window sized by its semi-major axis, which avoids a full sort or depth buffer over the map.

I.1.7 Confidence-based lifecycle

Measurement noise, imperfect poses and moved objects all leave surfels in the map that no longer correspond to any surface. We track a per-surfel confidence w, initialised to zero, and update it from three mutually exclusive pieces of per-frame evidence. Let $\Delta_i = z_i - D_t(\mathbf{u}_i)$ be the signed depth discrepancy of Eq. (9) for a surfel that projects inside the image. Then

$$w \leftarrow \begin{cases} \min(w + \gamma,\, w_{\max}) & |\Delta_i| < \tau \quad \text{(confirmed observation)}, \\ w - \lambda_{\text{free}} & \Delta_i < -\tau \quad \text{(observed in free space)}, \\ w - \lambda_{\text{miss}} & D_t(\mathbf{u}_i) \text{ invalid} \quad \text{(no sensor return)}, \\ w & \text{otherwise}, \end{cases} \tag{13}$$

and surfels with $w < 0$ are removed. $w_{\max} = 50$. The second case is the free-space constraint: the sensor has seen *through* the surfel, which is strong evidence that the surface is gone, and it is penalised an order of magnitude more heavily than a missing return ($\lambda_{\text{free}} = 10$ versus $\lambda_{\text{miss}} = \gamma = 1$). Surfels outside the frustum are never penalised, so geometry that leaves the field of view is retained indefinitely. Note that $\Delta_i > \tau$, a surfel occluded by a closer surface, carries no evidence either way and leaves w unchanged.

Confidence thus doubles as an implicit recency signal, and no per-surfel timestamp is required. Removal of pruned surfels and insertion of the newly spawned ones are applied as a single compaction of the map arrays at the end of the frame.