Resolution Where It Counts: Hash-based GPU-Accelerated 3D Reconstruction via Variance-Adaptive Voxel Grids

Lorenzo De Rebotti Emanuele Giacomini Giorgio Grisetti Luca Di Giammarino

Sapienza University of Rome, Italy

{derebotti,giacomini,grisetti,digiammarino}@diag.uniroma1.it

Abstract

Efficient 3D surface reconstruction from range data in real-time scenarios remains computationally and memory-intensive. Traditional volumetric methods using fixed grids or octrees are memory-inefficient, computationally expensive, and lack full GPU support. We present a variance-adaptive, multi-resolution voxel grid that adjusts resolution based on local SDF variance using a flat spatial hash table that enables constant-time access and full GPU parallelism. The system includes GPU-accelerated rendering via parallel quadtree-based Gaussian Splatting with adaptive splat density control. Our CUDA/C++ implementation achieves up to 5× speedup and 4× memory reduction versus fixed-resolution methods with only 3.3 % less accuracy on average.

1. Introduction

Accurate and scalable 3D reconstruction from depth or point cloud data is a core challenge in computer graphics, robotics, and vision. Applications such as robotic navigation, AR/VR, and large-scale mapping demand representations that are both geometrically precise and computationally efficient. Volumetric methods, particularly the Truncated Signed Distance Function (TSDF), are widely used for this purpose.

Introduced by Curless *et al.* [2], TSDF-based fusion integrates depth over time into a voxel grid, enabling robust surface reconstruction. KinectFusion [11] demonstrated real-time TSDF integration on dense, uniform voxel grids for small bounded scenes, while Voxel Hashing [12] addressed scalability by dynamically allocating voxel blocks using spatial hash tables, enabling real-time mapping on mobile platforms. Subsequent works, such as Voxblox [13] and Supereight [4], improved scalability through block-wise hashing and adaptive allocation strategies, prioritizing real-time performance often at the cost of geometric precision. Traditional approaches employ uniform grids, which scale poorly due to high memory and compute demands. To improve efficiency, adaptive structures like octrees and OpenVDB [10] have been proposed. OpenVDB provides sparse volumetric data structures through hierarchical trees, which are well-suited for 3D reconstruction in VDBFusion [18]. However, its multi-level hierarchy introduces lookup costs unsuitable for real-time integration. Octree-based methods like those by Funk. *et al.* [4] provide adaptive occupancy mapping but suffer from hierarchical access overhead, and they did not provide a GPU-compatible implementation. Recently, MAP-ADAPT [23] has explored adaptive resolution guided by semantics or image gradients, but it depends on structured RGB-D input and does not generalize to raw point clouds.

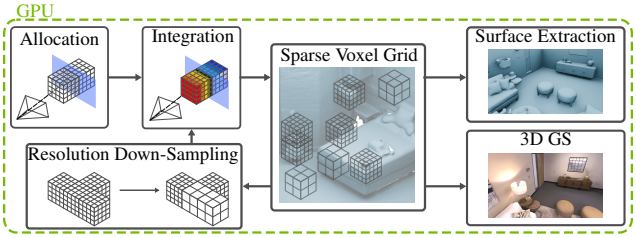

Figure 1. **System Overview:** Input data drives voxel block allocation and integration into a multi-resolution sparse voxel grid. Blocks with variance below a threshold are downsampled and re-integrated at coarser resolution. Our pipeline supports isosurface extraction or environment rendering.

In parallel, neural representations like PIN-SLAM [14] leverage hierarchical feature grids and MLP decoding for high-quality geometry reconstruction. While achieving accurate results, these approaches are typically limited to small-scale scenes and require substantial computing resources.

Recent methods, such as GSFusion [19], combine 3D Gaussian Splatting [5] with TSDF mapping, extending systems like Supereight for online rendering where the TSDF layer constrains Gaussian initialization and optimization.

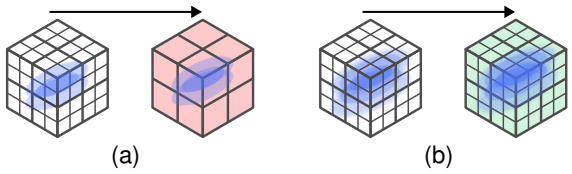

Figure 2. **Variance-driven merging.** (a) Low-variance blocks are reallocated at coarser resolution (red). (b) High-variance blocks remain fine (green).

However, their system requires frequent data transfers between the GPU and the CPU during rendering, which introduces performance bottlenecks.

We propose a variance-adaptive, multi-resolution voxel grid implemented on GPU using a flat spatial hash table. Our system adjusts resolution based on local TSDF variance, enabling fine detail where needed and coarse representation elsewhere, while achieving constant-time access and real-time performance across RGB-D and LiDAR inputs. Our contributions include:

- a variance-adaptive voxel grid with extended Marching Cubes [7] for seamless meshing across resolution boundaries
- a flat hash table managing mixed-resolution voxel blocks without hierarchical overhead
- a fully parallel GPU-based quad-tree for adaptive Gaussian Splatting

2. Technical Section

The TSDF map consists of spatially hashed voxels V_i with variable size $\nu \in \mathbb{R}^+$, chosen according to local voxel density. Voxels of equal size are grouped into blocks B_i, each identified by integer grid coordinates $\mathbf{b}_i = (x, y, z)^T \in \mathbb{Z}^3$. A voxel stores its signed distance D_i, confidence weight W_i, color, and variance σ_i^2 of the mean Signed Distance Function (SDF). At every frame k, sensor data (raw point cloud or depth image) is fused into this structure.

Integration follows standard volumetric fusion [2, 12]. Each voxel encodes a truncated signed distance D_i, positive in free space and negative behind surfaces, clipped to $\tau > 0$. For LiDAR sensors, rays are cast via DDA [1], updating voxels along the path to each point $\mathbf{p}$. For voxel center $\mathbf{x}$,

$$d_k(\mathbf{x}) = \text{clip}\Big(\frac{(\mathbf{p}-\mathbf{x})\cdot\mathbf{n}}{\|\mathbf{n}\|}, -\tau, \tau\Big), \tag{1}$$

with $\mathbf{n}$ the ray direction. For RGB-D cameras, depth values are back-projected with intrinsics $\pi^{-1}(u, v, d)$, and distances are computed along the viewing ray:

$$d_k(\mathbf{x}) = \text{clip}\big(\pi^{-1}(u, v, d) - \|\mathbf{x}\|, -\tau, \tau\big). \tag{2}$$

Voxel updates use a running weighted average:

$$D_i \leftarrow \tfrac{W_i D_i + w_k d_k(\mathbf{x})}{W_i + w_k}, \quad W_i \leftarrow W_i + w_k, \tag{3}$$

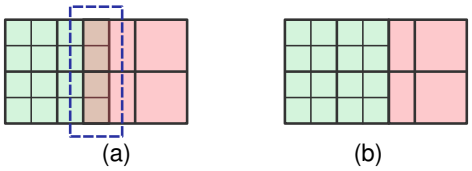

Figure 3. **Transition handling.** (a) Measurement-driven allocation creates overlaps between coarse and fine voxels. (b) Coarse voxels are truncated, aligning Marching Cubes vertices consistently.

with fixed $w_k = 1$ for consistent variance estimation. Using Welford's algorithm [20], we maintain voxel-level variance over time σ_i^2, which reflects geometric complexity: low in smooth areas, high near surfaces. This variance drives adaptive resolution by merging low-variance voxels into coarser blocks.

To address sparsity, voxel blocks are stored in a hash table [12]. Each block spans a fixed metric size but contains a resolution-dependent number of voxels. Positions $\mathbf{b}_i$ are mapped with a spatial hash:

$$H(x, y, z) = (xp_1 \oplus yp_2 \oplus zp_3) \bmod n_{hash}, \tag{4}$$

where $p_1 = 73856093$, $p_2 = 19349669$, $p_3 = 83492791$ [17]. Collisions are handled via per-entry buckets and linked lists. Unlike octrees or hybrid tree structures [4, 10], our unified hash table supports multiple resolutions in constant average time $O(1)$, avoiding costly recursive traversal and improving GPU parallelism.

Surface extraction is performed with Marching Cubes [7] on the multi-resolution grid. At boundaries between different resolutions, trilinear interpolation becomes ambiguous: a voxel corner at a coarse-fine interface may lack one or more neighbors at the expected resolution, resulting in undefined or inconsistent interpolated values. We address this by computing a weighted interpolation where contributions from finer voxels are prioritized, ensuring continuity and preserving geometric detail across resolution transitions. To prevent overlapping geometry across resolutions, coarse voxels adjacent to finer ones are truncated along shared faces (Fig. 3), following ideas from Transvoxel [6]. Finally, we collapse the vertices that are close to each other to ensure consistent meshing across scales.

Furthermore, the proposed adaptive grid is not limited to surface reconstruction; it also forms an efficient backbone for our high-quality, real-time rendering pipeline. To achieve this, we adopt a 3D Gaussian Splatting (3DGS) formulation [8], adapted for incremental processing of sequential images. Depth seeds Gaussians, while the voxel grid regulates their spawning. An image-space quadtree, built in parallel on the GPU, ensures density control. Contrast

ALGORITHM 1: Parallel GPU Quadtree Subdivision

Data: Image, Threshold, MinPixelSize
Result: FinalLeaves
NodeQueue ← RootNode covering full image;
while *NodeQueue not empty* **do**
  forall *Node ∈ NodeQueue (parallel block)* **do**
    Compute contrast with shared memory;
  end
  NewQueue ← ∅;
  forall *Node ∈ NodeQueue (parallel thread)* **do**
    if *Node.Contrast > Threshold and Node.Size > MinPixelSize* **then**
      Subdivide and append children to NewQueue;
    end
    else
      Add Node to FinalLeaves;
    end
  end
  NodeQueue ← NewQueue;
end
return FinalLeaves

$c(Q)$ for a quadtree Q is defined using luma-weighted RGB variance:

$$c(Q) = \mathbf{l} \frac{\sum_{\mathbf{q} \in Q} (\mathbf{q} - \text{mean}_{\text{rgb}}(Q))^2}{|Q|}, \tag{5}$$

with $\mathbf{l} = [0.2989, 0.5870, 0.1140]$. Subdivision continues until contrast falls below the threshold or pixel size is minimal. Unlike recursive CPU methods [19], our breadth-first GPU construction, in Alg. 1, exploits shared memory reductions, avoiding costly transfers and enabling high-throughput adaptive splat generation.

3. Experiments

To evaluate performance, we used several publicly available benchmarks: Replica (synthetic indoor) [15], Scannet (RGB-D indoor) [3], Newer College (LiDAR handheld) [21], and Oxford Spires (LiDAR handheld) [16].

Mapping quality is assessed with surface reconstruction metrics [9]: Accuracy (Acc), Completeness (Comp), Chamfer-L_1 (C-L_1), and F-score. Rendering quality is measured with Peak Signal-to-Noise Ratio (PSNR), Structure Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) [22]. For both mapping and rendering, we also reported Frame Per Seconds (FPS).

Concerning reconstruction, we compared our method against SOTA baselines: VDBFusion [18], VoxBlox [13], Supereight2 [4], and PIN-SLAM [14]. For rendering, we compared against GSFusion [19]. For both tasks, pipelines were run with ground-truth poses.

Experiments were executed on a PC (Intel i9-13900K, 128 GB RAM, RTX 4090). For single- and multi-resolution

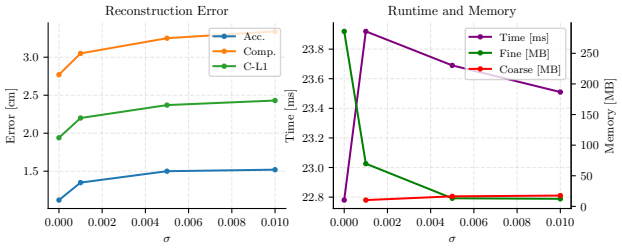

Figure 4. **Comparison of reconstruction error, runtimes, and memory usage for different variance thresholds σ.** The left plot reports accuracy, completeness, and Chamfer distance, while the right plot shows runtime and memory consumption of the fine- and coarse-level voxel structures.

grids, we used a block size of 512 (fine) and 64 (coarse), with hash parameters $p_1 = 73856093$, $p_2 = 19349669$, $p_3 = 83492791$.

Rendering used identical quadtree subdivision parameters (contrast 0.1, min pixel 1), and Gaussian optimization employed the same solver across methods, except Ours (+iterations), where we set more iterations to optimize GS parameters, using the single-resolution grid.

Our method achieves reconstruction quality comparable to or better than prior approaches across multiple datasets, with an accuracy loss of only 3.3 % on average, demonstrating that the variance-adaptive multi-resolution grid preserves accuracy while significantly reducing memory and computation requirements, as shown in Tab. 1.

Memory usage, in Tab. 2, shows that our multi-resolution configuration produces much more compact reconstructions than existing approaches, with reductions of 2.0–7.5× for RGB-D and 2.3–2.9× for LiDAR data, excluding VDBFusion, whose mesh size is slightly smaller. Runtime performance, reported in Tab. 2, demonstrates that our method is consistently faster than all baselines, achieving up to 13× speedup over state-of-the-art methods while maintaining high reconstruction fidelity. These gains arise from adaptive voxel allocation, which concentrates resolution in geometrically complex regions while coarsening uniform areas, a fully GPU-based pipeline that avoids CPU-GPU transfer overhead, and a flat hash table structure enabling constant-time access. Competing methods such as Supereight2 and Voxblox sometimes failed or required special configuration, highlighting the robustness, scalability, and efficiency of our approach across dense and sparse 3D data.

We evaluate rendering by integrating our voxel grid with a Gaussian Splatting renderer and comparing against GSFusion [19], see Tab. 3. Our multi-resolution design improves perceptual metrics (PSNR, SSIM, LPIPS), while the single resolution one achieves the highest rendering speed (28.05 FPS) thanks to a fully GPU-based quad-tree that dynamically manages splat density. Compared to GSFusion, which

Method	ScanNet (RGB-D)				Newer-College (LiDAR)				Avg.			
	Acc[cm]	Comp[cm]	C-L1[cm]	F-Score[%]	Acc[cm]	Comp[cm]	C-L1[cm]	F-Score[%]	Acc[cm]	Comp[cm]	C-L1[cm]	F-Score[%]
VDBFusion	3.336	**0.837**	2.086	92.696	**7.230**	**12.329**	**9.780**	**88.217**	4.504	**4.285**	4.394	91.352
PIN-SLAM	4.292	1.276	2.784	92.398	9.765	14.103	11.934	83.250	5.934	5.124	5.529	89.654
Voxblox	3.234	3.982	3.608	90.052	9.242	19.349	14.296	75.539	5.037	8.592	6.814	85.698
Supereight2 †	3.441	1.553	2.497	93.468	-	-	-	-	-	-	-	-
Supereight2 ‡	3.248	1.548	2.398	94.250	-	-	-	-	-	-	-	-
Ours	**2.030**	1.273	**1.637**	**96.845**	7.737	13.074	10.405	86.710	**3.743**	4.813	**4.267**	**93.804**
Ours (multi)	2.540	1.345	1.943	95.992	11.135	19.298	15.216	78.346	5.119	6.731	5.925	90.698

Table 1. **3D Reconstruction Results**. The RGB-D data are sequences of ScanNet [3] and LiDAR data are sequences of NC [21]. All the pipelines are run with ground truth fixed poses. Voxel size is set to 1 cm for RGB-D data and 20 cm for LiDAR data and the F-score is computed with a 10 cm error threshold for the RGB-D and 20 cm for the LiDAR. Best results are highlighted as **first** and second. Supereigth† represents single resolution grid, Supereight‡ represents multi-resolution grid. Both failed on all LiDAR sequences.

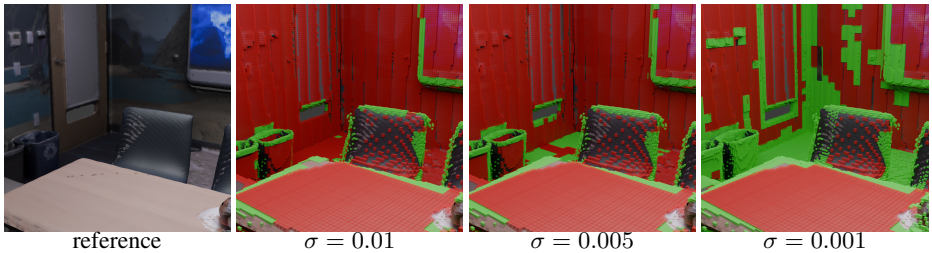

Figure 5. **Comparison of different variance thresholds σ.** In red, the coarser voxels, in green, the finer ones. From left to right: the reference ground truth mesh and the underlying grid at the respective thresholds. For simplicity, we show just two levels at a time. The sequence is part of the Replica dataset [15].

Method	Avg. Memory [MB] ↓	Avg. FPS ↑
VDBFusion	173.3	10.5
PIN-SLAM	361.5	12.4
Voxblox	212.2	8.6
Ours (single)	132.5	**50.9**
Ours (multi)	**88.7**	45.9

Table 2. **Memory and Runtime trade-off.** We report the average memory consumption [mb] and average FPS across all datasets. Best values are highlighted as **first** and second. Missing values ("–") indicate unavailable results.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	FPS ↑
GSFusion	34.65	0.949	0.056	14.99
Ours	33.90	0.949	0.057	**28.05**
Ours (+iterations)	34.27	0.951	0.052	14.22
Ours (multi)	**35.73**	**0.960**	**0.044**	23.70

Table 3. **Rendering quality and performance**. Comparison of perceptual metrics (PSNR, SSIM, LPIPS) and rendering speed (FPS) on Replica dataset [15]. Best results are highlighted as **first** and second.

uses Supereight2, our method delivers higher reconstruction quality and faster mapping time (14.99 FPS vs. 23.70 FPS), demonstrating both efficiency and fidelity.

Furthermore, we analyze the effect of the variance threshold σ on the trade-off between accuracy and memory. As shown in Fig. 4, lower thresholds improve reconstruction quality but increase fine voxel allocation, while higher thresholds reduce detail. Fig. 5 further illustrates that moderate σ values preserve quality while lowering memory usage.

4. Conclusion

We propose a multi-resolution voxel grid for real-time 3D reconstruction, adapting resolution via local TSDF variance and managing blocks with a flat hash table for constant-time GPU access. The method is sensor-agnostic, works with RGB-D and LiDAR inputs, and integrates with GPU Gaussian Splatting for high-quality rendering. Experiments show up to 4× memory savings, up to 5× speedup, and competitive accuracy.

Acknowledgments

This work has been supported by PNRR MUR project PE0000013-FAIR.

References

[1] John Amanatides and Andrew Woo. A Fast Voxel Traversal Algorithm for Ray Tracing. In *EG 1987-Technical Papers*. Eurographics Association, 1987. 2

[2] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, page 303–312, New York, NY, USA, 1996. ACM. 1, 2

[3] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2443, 2017. 3, 4

[4] Nils Funk, Juan Tarrio, Sotiris Papatheodorou, Marija Popović, Pablo F. Alcantarilla, and Stefan Leutenegger. Multi-Resolution 3D Mapping With Explicit Free Space Representation for Fast and Accurate Mobile Robot Motion Planning. *IEEE Robotics and Automation Letters*, 6(2): 3553–3560, 2021. 1, 2, 3

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1

[6] Eric Lengyel. Transition Cells for Dynamic Multiresolution Marching Cubes. *Journal of Graphics, GPU, and Game Tools*, 15(2):99–122, 2010. 2

[7] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, page 163–169, New York, NY, USA, 1987. ACM. 2

[8] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18039–18048, 2024. 2

[9] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4455–4465, 2019. 3

[10] Ken Museth. Vdb: High-resolution sparse volumes with dynamic topology. *ACM transactions on graphics (TOG)*, 32 (3):1–22, 2013. 1, 2

[11] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011. 1

[12] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32 (6):1–11, 2013. 1, 2

[13] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373. IEEE, 2017. 1, 3

[14] Yue Pan, Xingguang Zhong, Louis Wiesmann, Thorbjörn Posewsky, Jens Behley, and Cyrill Stachniss. PIN-SLAM: LiDAR SLAM Using a Point-Based Implicit Neural Representation for Achieving Global Map Consistency. *IEEE Transactions on Robotics*, 40:4045–4064, 2024. 1, 3

[15] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica Dataset: A Digital Replica of Indoor Spaces, 2019. 3, 4

[16] Yifu Tao, Miguel Ángel Muñoz-Bañón, Lintong Zhang, Jiahao Wang, Lanke Frank Tarimo Fu, and Maurice Fallon. The Oxford Spires Dataset: Benchmarking Large-Scale LiDAR-Visual Localisation, Reconstruction and Radiance Field Methods, 2024. 3

[17] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H. Gross. Optimized Spatial Hashing for Collision Detection of Deformable Objects. In *8th International Fall Workshop on Vision, Modeling, and Visualization, VMV 2003, München, Germany, November 19-21, 2003*, pages 47–54. Aka GmbH, 2003. 2

[18] Ignacio Vizzo, Tiziano Guadagnino, Jens Behley, and Cyrill Stachniss. VDBFusion: Flexible and Efficient TSDF Integration of Range Sensor Data. *Sensors*, 22(3):1296, 2022. 1, 3

[19] Jiaxin Wei and Stefan Leutenegger. GSFusion: Online RGB-D Mapping Where Gaussian Splatting Meets TSDF Fusion. *IEEE Robotics and Automation Letters*, 9(12):11865–11872, 2024. 1, 3

[20] B. P. Welford. Note on a Method for Calculating Corrected Sums of Squares and Products. *Technometrics*, 4(3):419–420, 1962. 2

[21] Lintong Zhang, Marco Camurri, and M. Fallon. Multi-Camera LiDAR Inertial Extension to the Newer College Dataset. *ArXiv*, 2021. 3, 4

[22] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 3

[23] Jianhao Zheng, Daniel Barath, Marc Pollefeys, and Iro Armeni. MAP-ADAPT: Real-Time Quality-Adaptive Semantic 3D Maps. In *Computer Vision – ECCV 2024*, pages 220–237, Cham, 2024. Springer Nature Switzerland. 1