LiftNav: Path Planning via Semantic Lifting in TSDF-Guided Gaussian Splatting

Hannah Schieber[1,*], Dominik Frischmann[1,*], Victor Schaack[1], Angela P. Schoellig[2] and Daniel Roth[1]

***Abstract*— Autonomous robots in unknown indoor environments require both reliable collision avoidance and object-level understanding. Classical representations such as TSDF support safe planning but lack semantics, while photorealistic methods like Gaussian Splatting (GS) provide rich appearance yet suffer from soft geometry, limiting precise obstacle avoidance. We present LiftNav, a hybrid navigation framework built on GSFusion's TSDF+GS dual map, augmented with a real-time pipeline of YOLO-based detection, TSDF-based 3D lifting, and B-spline trajectory optimization. This design enables flexible semantic navigation without dense 3D embeddings. We further introduce a hinge-loss-based collision penalty that improves trajectory smoothness and safety. We evaluate our approach in a simulation using the Replica dataset. Compared against a state-of-the-art radiance field baseline we show a 100% feasibility rate and shorter trajectories.**

I. INTRODUCTION

Robust navigation in complex 3D environments requires representations that capture geometry, appearance, and semantics, and potentially dynamics [1], [2], [3]. Radiance field methods, especially Gaussian Splatting (GS), enable efficient and photorealistic modeling, but remain weakly integrated with robotic pipelines, often limited to COLMAP-based setups [1]. Embedding semantics directly into GS improves scene understanding but increases computational cost and training time [4]. In contrast, combining accurate geometry with lightweight 2D semantic lifting offers a more efficient alternative.

We propose LiftNav, which combines photorealistic GS mapping with precise truncated signed distance function (TSDF)-based geometry, defers semantics to a lightweight 2D lifting stage, and restricts their use to navigation-relevant regions, enabling efficient and safe navigation under a lightweight policy. Our contributions are:

- A semantic lifting approach for TSDF-based GS.
- A robust and light-weight semantic navigation pipeline built on TSDF.

II. RELATED WORK

A. Hybrid Representations

ActiveGS [5] couples a GS map with a coarse voxel grid to enable spatial reasoning and exploration by targeting under-reconstructed regions. GSFusion [6] integrates TSDF geometry with GS for online RGB-D mapping, using geometric priors to regularize splatting and suppress artifacts.

B. 3D Path Planning within Radiance Fields

Leveraging high-quality GS scene representations for navigation is, for example, done by Adamkiewicz et al. [7] within neural radiance fields (NeRF). Within GS, Chen et al. introduced SplatNav [1], a real-time navigation pipeline in GS with two modules, *Splat-Plan* and *Splat-Loc*. Splat-Plan constructs safe convex corridors from Gaussian ellipsoids and generates smooth Bézier trajectories at $>$2 Hz, while Splat-Loc performs $\sim$25 Hz pose estimation by matching monocular RGB input to GS renderings via a Perspective-n-Point (PnP) formulation. However, they assume a high-quality known map, leaving a gap in scenarios requiring simultaneous exploration, mapping, and navigation. Similarly, GaussNav [8] operates on known envrionemnts. It uses a semantic GS, assigning 3D instance labels to Gaussians from clustered 2D masks. Target localization relies on exhaustive rendering and feature matching over candidate views, followed by 2D grid-based planning. While effective, this design incurs high computational cost and low update rates.

C. Semantic Perception and Spatial Projection

To leverage 2D semantic detections for navigation, the detections must be projected into metric 3D space. Lift, Splat, Shoot (LSS) [9] lifts pixels into 3D frustums via predicted depth and aggregates them into a unified bird eye view (BEV). In GS, semantics are often embedded into rendering [4], increasing training complexity. These methods typically rely on foundation models; while SAM lacks class labels, newer models like SAM3 [10] enable language-guided semantics. Lightweight models such as YOLOv8n-seg [11] offer real-time performance.

III. METHOD

A. Semantic Mapping

LiftNav extracts and localizes semantic targets in 3D through two primary phases: (i) 2D object detection [11] and (ii) spatio-temporal semantic lifting.

a) 2D Semantic Inference: For every frame, the detection model performs inference, outputting a set of bounding boxes $B = \{b_1, b_2, \ldots, b_n\}$. Each bounding box is associated with a semantic class label from the predefined language prompts and a corresponding confidence score.

* equal contribution

[1]Hannah Schieber, Dominik Frischmann, Victor Schaack, and Daniel Roth are with the Technical University of Munich, Human-Centered Computing and Extended Reality Lab, TUM University Hospital, Clinic for Orthopedics and Sports Orthopedics, Munich Institute of Robotics and Machine Intelligence (MIRMI) hannah.schieber@tum.de

[1]Angela Schoellig is with the Technical University of Munich, Learning Systems and Robotics Lab, Munich Institute of Robotics and Machine Intelligence (MIRMI)

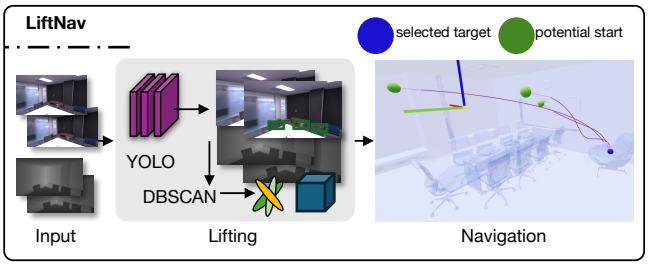


Fig. 1: Architecture. We process a RGB-D input stream via GSFusion [6], simultaneously we leverage YOLO [11] and employ our semantic lifting in clustering (center). Our path planning can leverage these semantic targets (right).

b) Semantic Lifting and 3D Projection: To transition from the image plane to the world coordinate system, we perform geometric back-projection. For each 2D detection, depth values are sampled from GSFusion's fused depth map [6]. Each pixel (u,v) within a detection box is mapped to a 3D point $\mathbf{p}_W$ in the world frame:

$$\mathbf{p}_W = \mathbf{T}_{WB}\left(D(u,v)\mathbf{K}^{-1}[u,v,1]^T\right)$$

where $D(u,v)$ is the depth value, $\mathbf{K}$ is the camera intrinsic matrix, and $\mathbf{T}_{WB}$ is the current estimated camera pose. To mitigate boundary depth outliers, we use the median 3D position of the masked region as the detection's centroid in the world frame. Operating online, the system observes objects from multiple viewpoints, accumulating a dense semantic point cloud over the trajectory.

c) Spatial Consolidation and Clustering: To mitigate projection errors, we apply Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [12] to cluster points into distinct object instances based on Euclidean distance. Sparse points are rejected as noise, preventing false positives. Each valid cluster is assigned a centroid and unique ID, forming 3D semantic waypoints for path planning.

B. Continuous-Time Trajectory Optimization

With 3D target coordinates established, our path planner generates collision-free trajectories within the safe, tangible geometry of the TSDF. We employ our LiftNav, building upon the B-Spline planner [7], [1]. We substantially modify this baseline to operate on explicit TSDF geometry rather than implicit radiance fields, and introduce custom boundary conditions to support robust semantic targeting.

a) Trajectory Parameterization: The trajectory is represented as a uniform B-Spline of degree p (typically $p = 8$), defined by a sequence of control points $\mathbf{Q} = \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_n\}$. This representation guarantees C^{p-1} continuity, ensuring that velocity, acceleration, and jerk are smooth. The local support of B-splines lets the optimizer adjust segments to avoid obstacles without affecting the global trajectory.

b) Robust Initialization via Discretized Search: To avoid local minima, we employ a hierarchical initialization. The environment is first discretized into a 3D occupancy grid with a conservative obstacle inflation layer of $1.5 \times r_{robot}$, where r_{robot} is the robot's physical radius. If a requested start or goal point lies within an inflated obstacle, a nearest-neighbor search projects the point to the closest safe navigable voxel. A 3D A* search is then executed between these safe projected voxels to find a collision-free "skeleton" path, which is resampled to seed the initial B-Spline control points.

c) Multi-Objective Trajectory Optimization: We formulate path planning as a constrained optimization problem, solved iteratively using the Adam optimizer and PyTorch's autograd engine. The objective function $J(\mathbf{Q})$ minimizes a weighted sum of four costs:

$$J(\mathbf{Q}) = w_{coll}L_{coll} + w_{dist}L_{dist} + w_{acc}L_{acc} + w_{jerk}L_{jerk}$$

For collision avoidance, we leverage the implicit distance information of the TSDF using a composite loss formulation:

$$L_{coll} = \sum\left[\max\left(0, 1 - \frac{d(\mathbf{p})}{m}\right) + \alpha\exp(-\beta d(\mathbf{p}))\right]$$

where $d(\mathbf{p})$ is the distance to the nearest surface sampled from the TSDF, and m is the safety margin (set to $1.5 \times r_{robot}$). The primary hinge loss component creates a potential field that aggressively pushes the path away from surfaces when they infringe upon the safety buffer.The secondary exponential term ($\alpha = 0.01, \beta = 2.0$) maintains a small, non-zero gradient outside the margin, ensuring continuous optimization. Path efficiency is enforced by L_{dist}, which minimizes the squared Euclidean distance between consecutive spline points. L_{acc} and L_{jerk} penalize the squared magnitudes of acceleration ($\ddot{\mathbf{p}}$) and jerk ($\dddot{\mathbf{p}}$), yielding graceful motion profiles that respect the actuators' dynamical limits.

d) Convergence and Finalization: While the initial A* seed is bounded by the safely projected free voxels, the continuous B-Spline optimization actively targets the *actual* requested start and goal coordinates. However, because semantic target coordinates inherently lie inside physical obstacle volumes (e.g., the geometric centroid of a designated object), allowing the continuous optimizer to reach the exact target would introduce inevitable collisions at the trajectory's extremities. To resolve this, the fully optimized B-Spline path is explicitly truncated at both ends by a distance equivalent to the inflation layer ($1.5 \times r_{robot}$). The endpoints of this safely cut spline are then linearly connected back to the exact start and goal coordinates. This formulation guarantees physical safety by preventing navigation inside the obstacle, while simultaneously mirroring the free-voxel projection mechanics of the baseline method (SplatNav) to ensure a fair and consistent comparison of path-length metrics.

IV. EVALUATION

We compare LiftNav against SplatNav [1] on Replica [13] in two settings. Scenario A, "Discretized Goal Benchmarking", using predefined handcrafted navigation start and goal. Scenario B, "Semantic Instance Targeting", assesses the fully integrated pipeline using YOLOv8n-seg model [11].

A. Implementation Details

We built upon GSFusion's [6] C++ implementation. Our semantic embedding enhanced detection model is exported

TABLE I: Comparative Benchmark: Safety, Efficiency, and Trajectory Quality across diverse indoor scenes.

Scene	Feasibility (%) ↑		Success (GT) (%) ↑		Time (s) ↓		Margin (m) ↑		Path Len (m) ↓		LE-Len (m) ↓		Max Jerk (m/s^3) ↓	
	SplatNav	*LiftNav*	*SplatNav*	*LiftNav*	*SplatNav*	*LiftNav*	*SplatNav*	*LiftNav*	*SplatNav*	*LiftNav*	*SplatNav*	*LiftNav*	*SplatNav*	*LiftNav*
office0	80±40	**100±0**	**100±0**	**100±0**	**0.15±0.03**	0.35±0.01	0.11±0.07	**0.11±0.04**	3.74±0.73	**2.77±0.81**	4.30±0.72	**3.45±0.82**	0.65±0.17	**0.09±0.05**
office2	**100±0**	**100±0**	**100±0**	**100±0**	**0.24±0.12**	0.40±0.14	**0.09±0.00**	-0.02±0.09	4.48±2.58	**3.40±1.48**	5.84±2.59	**4.07±1.52**	0.74±0.47	**0.11±0.03**
office3	0±0	**100±0**	N/A	**100±0**	N/A	**0.36±0.00**	N/A	**0.16±0.04**	N/A	**4.86±0.97**	N/A	**5.56±0.98**	N/A	**0.20±0.07**
office4	**100±0**	**100±0**	**100±0**	**100±0**	**0.27±0.06**	0.36±0.00	**0.10±0.07**	0.01±0.07	6.75±1.62	**4.85±0.92**	7.60±1.75	**5.53±0.92**	1.14±0.08	**0.17±0.05**
room0	**100±0**	**100±0**	**100±0**	**100±0**	**0.36±0.08**	**0.36±0.01**	**0.13±0.06**	0.10±0.07	5.98±1.39	**4.57±0.89**	6.79±1.35	**5.24±0.91**	0.76±0.07	**0.21±0.09**
room1	**100±0**	**100±0**	**100±0**	60±49	**0.26±0.09**	0.35±0.00	**0.09±0.04**	0.02±0.09	3.85±1.59	**2.86±1.10**	4.57±1.56	**3.52±1.12**	0.43±0.10	**0.11±0.02**
room2	**100±0**	**100±0**	**100±0**	**100±0**	**0.24±0.05**	0.35±0.00	0.10±0.09	**0.11±0.11**	4.41±1.14	**3.78±1.24**	5.27±1.30	**4.45±1.26**	0.58±0.10	**0.13±0.06**
AVERAGE	83	**100**	100*	**94**	**0.25**	0.36	**0.10**	0.07	4.87*	**3.87**	5.73*	**4.55**	0.72*	**0.15**

**SplatNav averages exclude office3 where the planner failed to find a valid trajectory. Success rate is calculated based on feasible paths.*

TABLE II: Semantic Instance Targeting: Safety, Efficiency, and Trajectory Quality.

Scene	Feasibility (%) ↑		Success (GT) (%) ↑		Time (s) ↓		Margin (m) ↑		Path Len (m) ↓		LE-Len (m) ↓		Max Jerk (m/s^3) ↓	
	SplatNav	*LiftNav*	*SplatNav*	*LiftNav*	*SplatNav*	*LiftNav*	*SplatNav*	*LiftNav*	*SplatNav*	*LiftNav*	*SplatNav*	*LiftNav*	*SplatNav*	*LiftNav*
office0	44±50	**100±0**	**94±24**	90±30	**0.14±0.04**	0.37±0.01	**0.13±0.08**	0.01±0.06	2.98±1.03	**2.00±0.70**	3.74±1.08	**2.65±0.71**	0.59±0.15	**0.10±0.07**
office2	86±35	**100±0**	**100±0**	76±43	**0.16±0.07**	0.36±0.01	**0.30±0.07**	-0.00±0.08	2.85±1.27	**2.26±1.02**	3.76±1.27	**2.90±1.03**	0.64±0.30	**0.11±0.05**
office3	41±49	**100±0**	**100±0**	59±49	**0.09±0.03**	0.36±0.01	**0.21±0.10**	0.05±0.05	**1.21±0.68**	2.48±1.28	3.88±0.57	**3.14±1.30**	0.33±0.30	**0.12±0.08**
office4	**100±0**	**100±0**	**100±0**	88±33	**0.15±0.05**	0.36±0.01	**0.20±0.09**	0.03±0.08	3.58±1.34	**2.34±0.97**	4.38±1.41	**2.99±0.99**	1.03±0.21	**0.10±0.05**
room0	92±28	**100±0**	**100±0**	81±40	**0.21±0.08**	0.36±0.01	**0.17±0.09**	0.02±0.05	3.36±1.42	**2.29±1.09**	4.08±1.46	**2.94±1.11**	0.63±0.17	**0.09±0.05**
room1	**100±0**	**100±0**	**100±0**	93±25	**0.20±0.07**	0.36±0.01	**0.20±0.09**	-0.01±0.11	3.06±1.19	**2.01±0.84**	3.88±1.28	**2.67±0.86**	0.57±0.15	**0.09±0.05**
room2	**100±0**	**100±0**	**100±0**	67±47	**0.15±0.04**	0.37±0.02	**0.18±0.07**	-0.03±0.11	**2.59±0.85**	2.73±2.26	3.44±0.94	**3.41±2.33**	0.57±0.12	**0.15±0.10**
AVERAGE	80	**100**	**99**	79	**0.16**	0.36	**0.20**	0.01	2.80	**2.30**	3.88	**2.96**	0.62	**0.11**

to the ONNX format and executed via the OpenCV DNN module utilizing a CUDA backend.

a) Hyperparameters: We set B-Spline degree $p = 8$, exponential decay loss for collision avoidance, collision penalty weight $w_{coll} = 2000$, distance penalty weight $w_{dist} = 1.0$ (normalized by path length), acceleration penalty weight $w_{acc} = 0.5$, jerk penalty weight $w_{jerk} = 0.5$, and a simulated robot diameter of 0.2 m.

b) Baselines: For the baseline (SplatNav), which relies on Nerfstudio [14] to generate its GS representation, we processed the Replica dataset with their pipeline.

c) Metrics: We measure total wall-clock time (dataset processing (raw RGB-D streams) to the final data structures), performing three repetitions per scene to account for computational variance. We quantify performance across three axes: safety (Success Rate ground truth (GT), *GS Safety Margin*), navigation efficiency (*Path Length (L)*, *Line-of-Sight Path Length (L_{LE})*), and trajectory quality (*Average Jerk*, *Maximum Jerk*).

d) Hardware: All experiments were performed on a single workstation equipped with an NVIDIA RTX 3090 GPU and an AMD Ryzen 7 7700X processor.

B. Comparative Benchmark Scenarios

1) Reconstruction Pipeline Efficiency: Our underlying baseline GSFusion [6] demonstrates a clear efficiency advantage, outperforming Nerfstudio's mean reconstruction time in four of the seven evaluated scenes and achieving a faster overall average (439.44s vs. 443.30s). Nerfstudio outputs solely produce GS, forcing SplatNav to subsequently conduct expensive convex hull derivations and spatial decompositions during the active planning phase. In contrast, GSFusion simultaneously reconstructs both the GS and the dense, obstacle-aware TSDF.

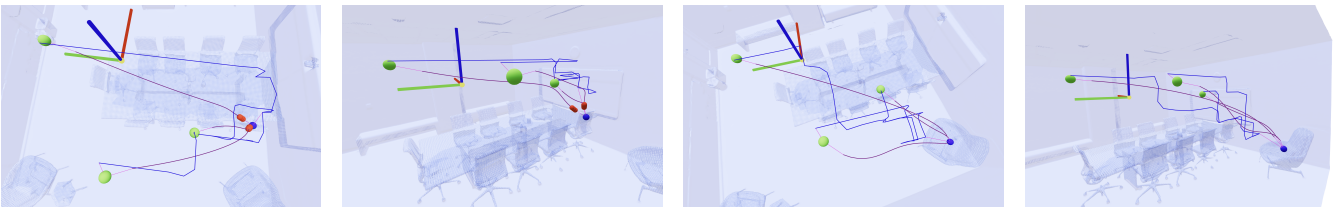
2) Scenario A: Discretized Goal Navigation: SplatNav holds a slight edge in computation time (0.25s vs 0.36s) due to its convex Safe Flight Corridor (SFC) generation (Table I). However, LiftNav compensates for this with shorter and more direct routes (Average LE-Len: 4.55m vs 5.73m). The smoother, shorter paths are evident on visual inspection (Fig. 3).

3) Scenario B: Semantic Instance Targeting: Navigating directly to semantic objects introduces the "Semantic Trap": goal centroids often lie flush against a physical mesh (e.g., a television screen or vase). As seen in Table II, SplatNav's SFCs struggles immensely in these scenarios, dropping to 44% and 41% feasibility in *office0* and *office3*, respectively, as it cannot mathematically generate a collision-free corridor directly into a surface boundary. LiftNav, conversely, manages to achieve 100% feasibility rate for semantic targeting. To reach these flush targets, our planner naturally dips into the defined safety margin at the very end of the trajectory, resulting in average margins near zero (0.01m) and triggering technical GT collisions (dropping the strict success rate to 79%) (Fig. 2). Despite this strict geometric penalty at the target node, the overarching trajectory quality remains dominant (Fig. 2). LiftNav consistently generates shorter paths (2.96m vs 3.88m LE-Len) and maintains an ultra-smooth Maximum Jerk profile (0.11 m/s^3 vs 0.62 m/s^3), demonstrating its reliability and grace in fully integrated, vision-driven navigation tasks.

V. DISCUSSION

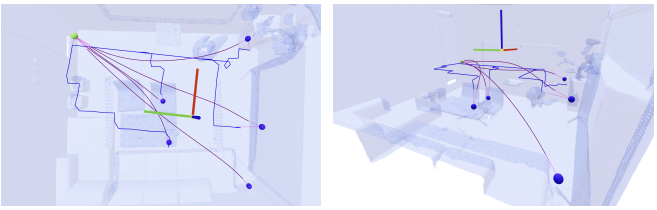
While SplatNav achieved faster planning times (~ 0.25s vs. 0.36s), this speed comes at a severe cost to trajectory quality. LiftNav's C^5 continuous B-Spline parameterization naturally regularizes the motion, suppressing peak jerk to an average of just 0.15 m/s^3.

The absolute 100% feasibility rate of LiftNav highlights the resilience of gradient-based collision avoidance. In environments like *office3*, SplatNav failed due to path initialization and subsequent optimization failures. LiftNav bypasses this limitation, leveraging our proposed Hinge-Loss formulation as a non-vanishing restorative force, allowing the Adam optimizer to smoothly slide control points around obstacles without becoming trapped by the topological constraints that



(a) Top-down view (b) Side view (c) Top-down view (d) Side view

Fig. 2: Planning results of office4 for a semantic target from the reconstruction showing collisions near the target (left, left center) and collision-free (right center, right) (SplatNav (blue), LiftNav (purple), start (green), target (blue), collisions (red).



(a) Top-down view (b) Side view

Fig. 3: Planning results of office0 for handcrafted targets[1].

might paralyze SFC methods.

Scenario B shows that SplatNav struggles with semantic objects, with feasibility dropping (e.g., 41% in *office3*) since it cannot generate a collision-free corridor into a solid boundary. LiftNav succeeded in 100% of these tasks. However, the data shows that to reach these targets, the optimizer deliberately forces the terminal segment of the spline into the safety margin, resulting in GT collisions (dropping the strict success rate to 79%) (see Section VI).

VI. Limitation and Future Work

Boundary violations originate upstream, not in the planner. Multi-view semantic lifting places target centroids deep inside objects, especially for larger objects and diverse views, forcing the optimizer to route the spline through the safety margin. This trade-off, is a design choice to focus first on efficient target oriented path planning while future work will focus on improving in-flights safety and collision avoidance.

Currently, we only evaluate on the Replica dataset. Future work shall move to more challenging datasets and outdoor scenarios.

VII. Conclusion

LiftNav introduces a hybrid navigation framework that bridges collision-safe geometric planning and flexible semantic reasoning. Built on GSFusion, it integrates semantic lifting to localize and navigate to semantic targets without dense 3D semantic mapping. Trajectory generation is formulated as continuous B-spline optimization over TSDF gradients, yielding smoother and more feasible motion than discretized SFC. A hinge-loss collision term further improves the safety and efficiency trade-off by enabling smooth navigation through complex environments. The framework demonstrates strong robustness in constrained semantic trap scenarios where SFC methods fail, providing topologically consistent and dynamically feasible trajectories.

Acknowledgments

This work was partially supported by the Technical University of Munich, Munich, Germany, through its MIRMI Seed Fund program, and by the Robotics Institute Germany (RIG) under grant 16ME0997K.

References

[1] T. Chen, O. Shorinwa, J. Bruno, A. Swann, J. Yu, W. Zeng, K. Nagami, P. Dames, and M. Schwager, "Splat-nav: Safe real-time robot navigation in gaussian splatting maps," *IEEE Transactions on Robotics*.

[2] L. Brunke, Y. Zhang, R. Römer, J. Naimer, N. Staykov, S. Zhou, and A. P. Schoellig, "Semantically safe robot manipulation: From semantic scene understanding to motion safeguards," *IEEE Robotics and Automation Letters*, 2025.

[3] B. Bogenberger, O. Harrison, O. Dahanaggamaarachchi, L. Brunke, J. Qian, S. Zhou, and A. P. Schoellig, "Where did i leave my glasses? open-vocabulary semantic exploration in real-world semi-static environments," *IEEE Robotics and Automation Letters*, 2026.

[4] H. Schieber, D. Frischmann, S. Boche, V. Schaack, A. Schoellig, S. Leutenegger, and D. Roth, "Core-gs: Coarse-to-refined gaussian splatting with semantic object focus," 2025. [Online]. Available: https://arxiv.org/abs/2509.04859

[5] L. Jin, X. Zhong, Y. Pan, J. Behley, C. Stachniss, and M. Popović, "Activegs: Active scene reconstruction using gaussian splatting," *IEEE Robotics and Automation Letters*, 2025.

[6] J. Wei and S. Leutenegger, "Gsfusion: Online rgb-d mapping where gaussian splatting meets tsdf fusion," *IEEE Robotics and Automation Letters*, vol. 9, no. 12, pp. 11 865–11 872, 2024.

[7] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-only robot navigation in a neural radiance world," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4606–4613, 2022.

[8] X. Lei, M. Wang, W. Zhou, and H. Li, "Gaussnav: Gaussian splatting for visual navigation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 5, pp. 4108–4121, 2025.

[9] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," in *European conference on computer vision*. Springer, 2020, pp. 194–210.

[10] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Suris, C. Ryali, K. V. Alwala, H. Khedr, A. Huang *et al.*, "Sam 3: Segment anything with concepts," *arXiv preprint arXiv:2511.16719*, 2025.

[11] G. Jocher, J. Qiu, and A. Chaurasia, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[12] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[13] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*.

[14] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja *et al.*, "Nerfstudio: A modular framework for neural radiance field development," in *ACM SIGGRAPH 2023 conference proceedings*, 2023, pp. 1–12.