City-scale Motion Planning in Radiance Fields for Generating Virtual Walk-Throughs

Marit Zenker, Ethan Fuchs, Lukas Willmann, Maja Temerinac-Ott, and Uwe Hahne

Hochschule Furtwangen
firstname.lastname@hs-furtwangen.de

Abstract. In this work, we explore the deployment of city-scale 3D Gaussian Splatting (3DGS) scenes for the creation of visually appealing virtual routes. We present an automated virtual route planner that creates smooth, obstacle-free paths between user-defined start and end points. These paths can be used to create first-person walk-throughs rendered within the 3DGS environment, enabling intuitive exploration of complex environments. We demonstrate the effectiveness of our implementation across various test scenes, highlighting its ability to consistently generate collision-free and smooth paths. We evaluate eight path planning algorithms on two 3DGS scenes considering success rate (SR), path planning time and path length. The results show that our system achieves a 100% SR across two different test scenes, matching or exceeding the performance of state-of-the art methods.

Keywords: 3D Gaussian Splatting; Motion and Path Planning; Scene Reconstruction and Rendering

1 Introduction

Navigating unfamiliar environments such as university campuses, airports, hospitals, or museums can be disorienting for newcomers. Complex layouts with multiple buildings and outdoor spaces make it difficult to find the right path. Traditional maps provide only top-down perspectives, often lacking the visual context needed for intuitive navigation. Systems like Google Street View provide realistic street-level imagery, but the views are not seamlessly connected. Transitions between these views are abrupt, and the reliance on 2D geometry reduces the overall sense of immersion.

To address these limitations, we propose a virtual route planning system that enables users to explore an environment through an immersive, high-fidelity 3D representation. By generating smooth and realistic first-person camera paths, users can virtually experience the space. Developing this system requires a computationally efficient yet detailed 3D scene representation and a method for generating smooth camera paths. We achieve this by combining 3D Gaussian Splatting (3DGS) [1] with motion planning techniques. The main contributions of this work are first, a new approach to motion planning by generating a voxel-based simplification of 3DGS scenes to enable automated planning of collision-free paths. Second, we provide an elaborate evaluation of existing path planning algorithms as well as the efficiency and quality of generated camera paths. Those generated paths allow the rendering of immersive first-person views at interactive rates to visualize realistic, high-fidelity virtual walkthroughs. This work is a next step towards immersive virtual navigation systems and demonstrates the potential of using 3DGS scenes for motion planning.

2 Related Work

3D Gaussian Splatting (3DGS) [1] is a recent method for creating and rendering radiance fields from a set of 2D images. Scenes are represented as sets of explicit 3D Gaussians, enabling high-quality synthesis of novel views. It has also been demonstrated that 3DGS can be scaled to large city-level scenes through hierarchical structures [2] and enhanced for urban and street environments [3,4], reinforcing the relevance of motion planning using such 3D scene representations for various future applications.

2.1 Motion planning

Motion planning can classically be described as the problem of moving an object from one point to another in a given 3D environment. It is used across diverse fields, from video games and molecular simulations to robots e.g. autonomous vehicles, leading to a wide range of algorithms tailored to different scenarios [5]. Notable examples include Rapidly-exploring Random Trees (RRT)[6] and Probabilistic Roadmaps (PRM)[7], along with their many extensions [8,9,10]. Despite this progress, to the best of our knowledge, only a few studies have explored integrating these planners with 3DGS as a scene representation.

Some studies have explored the use of 3DGS in robotics, particularly in the context of Simultaneous Localization and Mapping (SLAM) [11,12,13]. In addition to SLAM, there has been research on route planning and navigation, such as *Splat-Nav* [14], *GaussNav* [15], and *GS-Planner* [16]. *Splat-Nav* [14] is a real-time navigation pipeline for robots based on a pre-trained, static 3DGS scene. It combines two modules: *Splat-Loc* for localization and *Splat-Plan* for path planning. *Splat-Plan* creates a 3D binary occupancy grid from the scene, then uses the A* algorithm [17] to generate candidate paths. An ellipsoid-based collision check filters out unsafe paths by comparing the robot's shape with nearby 3D Gaussians. Once a valid path is found, it is refined into a smooth, feasible trajectory using B-spline optimization. *GaussNav* [15] is a system for Instance-Image Goal Navigation (IIN), where the agent must locate an object shown in a goal image. The system incrementally builds a 3DGS scene representation while exploring its environment. It partitions the scene into subsets containing geometric and semantic information, then generates a 2D bird-eye view grid map for planning using the fast marching method [18]. To improve performance, it uses isotropic 3D Gaussians and view-independent color, limiting the rendering quality. *GS-Planner* [16] focuses on high-quality scene reconstruction through active exploration. It employs RRT* [6] to sample viewpoints and introduces a differentiable cost to penalize viewpoints too close to 3D Gaussians based on distance, opacity, and radius. Axis-Aligned Bounding Box (AABB) filtering improves efficiency by limiting collision checks to obstacles near the path. However, *GS-Planner* also uses isotropic 3D Gaussians, resulting in a reduced rendering quality.

These systems focus on robot motion planning, often trading visual quality for efficiency or relying on 2D planning unsuitable for complex environments. They are also designed specifically for robots rather than camera paths, limiting their applicability to our use case. Therefore, we developed our own solution and focus on evaluating its path planning performance along various established methods.

3 Implementation

3.1 Problem Statement

Given a 3DGS scene and user-defined start and end coordinates, our system generates a human-navigable path between these points that avoids obstacles, maintains a realistic walking height above the ground, and avoids steep slopes.

A virtual camera then follows this path to render a first-person video simulating real-world traversal. To ensure comfort and immersion, the camera motion must meet four criteria: `Smoothness` (no abrupt direction changes; minimal sharp turns), `Speed` (consistent movement speed without sudden changes), `Orientation` (always face the direction of travel, with a level horizon), `Position` (maintain a fixed height above ground, a safe distance from obstacles, and avoid steep inclines or declines).

Our system works best on high-quality 3DGS scenes because artifacts like *floaters* (misplaced 3D Gaussians) and holes might lead to unrealistic paths. Path generation is tailored for smooth camera motion and realistic human navigation. More advanced features, such as handling dynamic obstacles, real-time replanning, or supporting alternative agents like robots or drones, are currently out of scope.

Designed for consumer-grade hardware, the system runs efficiently on mid-range GPUs and supports real-time path generation. Potential use cases include, but are not limited to: route planning, virtual tours, accessibility-aware navigation, games and simulations, and augmented reality applications.

3.2 Processing

Figure 1 illustrates the pipeline of our system. The input is a 3DGS scene, which is converted to a voxel grid representation. To ensure the path avoids collisions with obstacles in the 3DGS scene, collision detection is necessary. While ellipsoid intersection based on 3D Gaussians is possible, it is computationally expensive. Instead, we convert the 3DGS scene into a voxel grid for efficient collision checking. First, the voxel size is determined

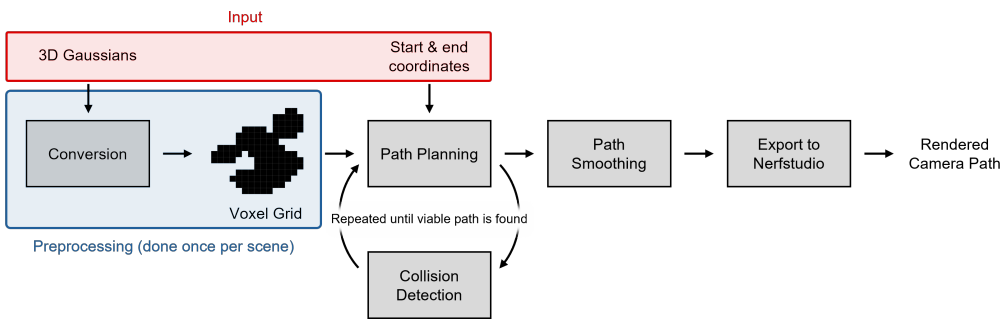


Fig. 1. Our pipeline. The 3DGS scene is converted to a voxel grid for collision detection. The camera path is generated using *OMPL* and smoothed before being exported to *Nerfstudio* for rendering.

based on either the average size of all 3D Gaussians or by manual input. The scene is

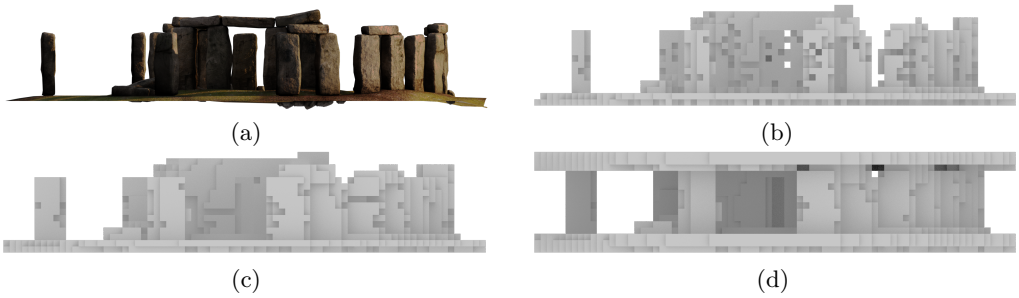


Fig. 2. The original *Stonehenge* mesh (a) and its corresponding voxel grid (b). Voxel grid with added padding (c) and height constraints (d).

then divided into a grid of voxels, with each voxel representing a small part of the environment. Finally, each 3D Gaussian's position is used to mark the corresponding voxel as occupied. This step is illustrated in Figure 2.

To fill gaps and maintain a safe distance from obstacles, we pad the voxel grid by marking all voxels within a specified horizontal range of occupied voxels as occupied too (Figure 2 (c)). To keep the path at a consistent distance from the ground and avoid areas without valid ground support, such as gaps or out-of-bounds regions, we mark voxels without valid ground beneath them as occupied (Figure 2 (d)).

We represent the scene using a sparse voxel grid implemented as a Python dictionary, storing only the occupied voxels. This approach significantly reduces memory usage and accelerates collision detection by limiting checks to relevant voxels. Using this voxel grid, the system generates a path connecting the start and end points provided by the user. The path is generated using the *Open Motion Planning Library (OMPL)* [19] and considers the voxel grid scene representation to ensure the path is collision-free. Then, the generated path is smoothed to ensure a pleasant camera movement. Finally, the path is exported to *Nerfstudio* [20] for visualization and rendering.

3.3 Dataset

We evaluated our system on two different scenes with varying sizes and difficulties: *Stonehenge* [14] and *Small City* [1]. *Stonehenge* (see Fig. 2(a)) is a synthetic dataset and includes a polygon mesh from which we generated 200 images used to train a 3DGS scene using *Nerfstudio* [20]. Its familiarity and moderate number of obstacles make it well suited for evaluating motion planning methods.

In contrast, *Small City* was created from real-world data using 5822 images. We use this scene to test our system's performance on a larger environment. It depicts an urban building block, streets, and trees, and covers an area of $40,000m^2$. It was created by taking images from 6 cameras at different angles while driving around a block of buildings. An impression of the scene as well as the route taken during image capture is shown in Figure 3.

3.4 Path planning

To evaluate the path planning, we use the *OMPL* library [19] with a 3D real vector state space ($\mathbb{R}^3$) and a custom state validity checker based on the voxel grid. A state is valid if



Fig. 3. The *Small City* scene. The left image shows a frame rendered from the 3DGS scene, while the right image displays an overhead point cloud view with the camera path during image capture highlighted in green. The red frustum indicates the camera's current position and orientation corresponding to the rendered frame on the left [2].

its voxel is unoccupied, which inherently enforces padding and height constraints. Additionally, we ensure the slope between consecutive states stays below a set threshold. Start and goal positions are given as 3D coordinates, and an appropriate planner computes the path within a specified time limit.

We evaluated both non-optimizing and optimizing planners. Non-optimizing planners return the first valid path found, offering fast but potentially suboptimal results. Optimizing planners continue refining the path within a given time limit to improve quality based on a user-defined objective (e.g. minimizing path length) [19].

Starting from the initial state, the planner proposes new states and connects them to the previous one. Each segment of the path is sampled to check for collisions. If any sampled state along the segment is invalid, the entire segment is discarded, and a new proposal is generated beginning from the last valid state. This process repeats until a collision-free path to the goal is found. To ensure smooth, visually pleasing camera movement, we refine the path using path simplification and B-Splines.

Finally, the path is used as input to render a video from a virtual camera moving through the scene. To ensure consistent movement speed, the path is resampled so that the states are evenly spaced. Since the states specify only the camera's position, we compute an optimal camera orientation based on previous positions ensuring the camera smoothly faces the direction of motion. This allows us to render an animation along the camera path using the states as keyframes.

4 Results

We consider various metrics to evaluate the performance of our path planning system. The success rate (SR) is defined as the percentage of successful path generation attempts. A path is considered successful if it connects the start and end points without colliding with obstacles or violating any constraints. The computation time to generate a path

between the start and end points should be as low as possible to provide a responsive user experience. Further, a short path length is generally preferable, as it minimizes the time required to traverse it.

We evaluated five non-optimizing and three optimizing path planning algorithms. We used a voxel resolution of 115×110×24 with a padding of 1 and minimum and maximum distances to the ground of 3 and 5 voxels, respectively. The planner range was set to 0.1, the state validity checking resolution to 0.001, and the maximum time given for each path was 5 seconds. Each set of points was used to generate 100 paths per planner, resulting in a total of 900 paths per planner. For each planner, we evaluated its success rate, computation time, and path length. The results for both data sets are given in Table 1 and 2.

Table 1. Results of all path planning algorithms tested on the *Stonehenge* scene.

Category	**Planner**	**SR (%) ↑**	**Mean Computation Time ± SD (s) ↓**	**Mean Path Length ± SD (m) ↓**
Non.	RRT [6]	100.00	0.062 ± 0.036	44.937 ± 4.480
Non.	**RRTConnect** [8]	100.00	**0.030 ± 0.017**	45.237 ± 5.390
Non.	PDST [21]	81.44	0.402 ± 0.535	44.907 ± 5.184
Non.	STRIDE [22]	99.89	0.227 ± 0.216	49.608 ± 8.053
Non.	PRM [7]	100.00	0.134 ± 0.056	42.159 ± 2.804
Opt.	**RRT*** [10]	100.00	5.033 ± 0.010	**37.656 ± 0.121**
Opt.	RRTX [23]	99.89	5.049 ± 0.121	37.816 ± 0.178
Opt.	BIT* [24]	100.00	5.098 ± 0.419	37.699 ± 0.144

For each scene different planners get the best result. The difference between optimizing and non-optimizing planners becomes apparent when comparing path lengths: All optimizing planners have shorter average path lengths with a much lower SD than non-optimizing planners, sometimes even with significant differences. However, the non-optimizing planners are much faster for both scenes.

We evaluated the path planning algorithms on *Small City* by generating 100 paths per planner, using start and goal points located at opposite corners of the environment. The scene was voxelized at a resolution of 169 × 196 × 59, incorporating a padding of 1 and a minimum and maximum ground distance of 0 and 20, respectively. The maximum allowed time per path was set to 10 seconds, with a planner range of 0.7 and a state validity checking resolution of 0.01.

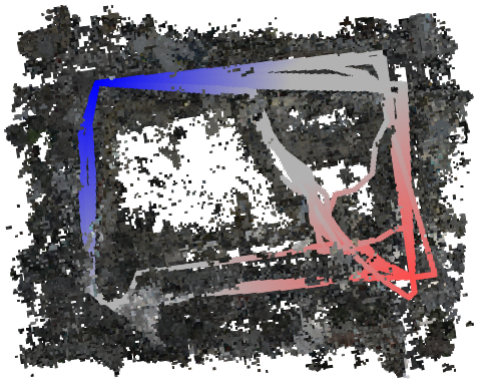
The computation time is generally longer than in the *Stonehenge* scene, but still under one second for nearly all non-optimizing planners. The average path lengths are relatively similar for all planners, even across both categories. This is because the free space only allows for a very limited number of paths. Figure 4 shows the paths generated by a non-optimizing and an optimizing planner.

5 Conclusion

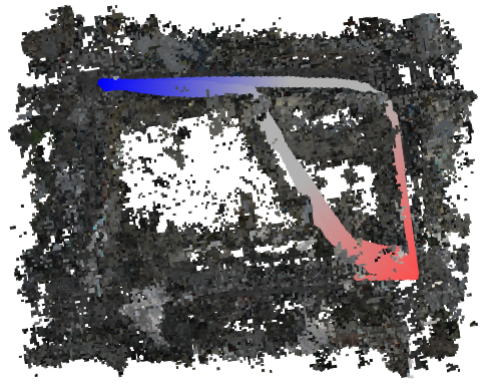
In this work, we introduced a method for collision-free path planning using a 3DGS scene representation. For this, we developed a pipeline that converts a given 3DGS scene into a voxel grid and applies motion planning to generate a path that can be rendered in *Nerfstudio* for a high-quality first-person view of the scene along the path. We evaluated

Table 2. Results of all path planning algorithms tested on the *Small City* scene.

Category	Planner	SR (%) ↑	Mean Computation Time ± SD (s) ↓	Mean Path Length ± SD (m) ↓
Non.	RRT [6]	100.00	0.735 ± 1.278	9090.055 ± 195.139
Non.	RRTConnect [8]	100.00	0.433 ± 0.622	9072.399 ± 286.758
Non.	**PDST** [21]	100.00	**0.066 ± 0.038**	9357.820 ± 398.638
Non.	STRIDE [22]	2.00	7.904 ± 0.394	8521.576 ± 423.983
Non.	PRM [7]	100.00	0.145 ± 0.078	9218.557 ± 372.155
Opt.	RRT* [10]	100.00	10.089 ± 0.033	9120.528 ± 243.071
Opt.	RRTX [23]	98.00	10.048 ± 0.037	8795.128 ± 277.223
Opt.	**BIT*** [24]	100.00	10.927 ± 2.125	**8129.309 ± 448.369**



(a) 100 paths generated by RRTConnect.



(b) 100 paths generated by RRTX.

Fig. 4. Comparison of non-optimizing vs. optimizing planners in the *Small City* scene. The start and goal points are marked in blue and red, respectively.

this system on two different test scenes, showing that it can generate paths with a success rate of 100% in scenes of varying sizes and complexities as well as scenes created from synthetic and real-world data. In the future, we plan to embed large language model support in the 3DGS scene similar to LEGaussians [25]. This would make the system more user-friendly as navigation tasks could be given in natural language.

References

1. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering (2023)
2. Kerbl, B., Meuleman, A., Kopanas, G., Wimmer, M., Lanvin, A., Drettakis, G.: A hierarchical 3d gaussian representation for real-time rendering of very large datasets. ACM Transactions on Graphics **43**(4) (July 2024)
3. Yan, Y., Lin, H., Zhou, C., Wang, W., Sun, H., Zhan, K., Lang, X., Zhou, X., Peng, S.: Street gaussians: Modeling dynamic urban scenes with gaussian splatting. In Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., Varol, G., eds.: Computer Vision – ECCV 2024, Cham, Springer Nature Switzerland (2025) 156–173
4. Zhou, X., Lin, Z., Shan, X., Wang, Y., Sun, D., Yang, M.H.: Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In: Proceedings of

the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (June 2024) 21634–21643
5. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge, U.K. (2006) Available at http://planning.cs.uiuc.edu/.
6. LaValle, S.M.: Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98–11, Computer Science Department, Iowa State University (October 1998) https://cs.stanford.edu/people/aydin/RRT/TR98-11.pdf.
7. Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation **12**(4) (1996) 566–580
8. Kuffner, J., LaValle, S.: Rrt-connect: An efficient approach to single-query path planning. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065). Volume 2. (2000) 995–1001 vol.2
9. Bohlin, R., Kavraki, L.: Path planning using lazy prm. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065). Volume 1. (2000) 521–528 vol.1
10. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning (2011)
11. Keetha, N., Karhade, J., Jatavallabhula, K.M., Yang, G., Scherer, S., Ramanan, D., Luiten, J.: Splatam: Splat, track & map 3d gaussians for dense rgb-d slam (2024)
12. Yugay, V., Li, Y., Gevers, T., Oswald, M.R.: Gaussian-slam: Photo-realistic dense slam with gaussian splatting (2024)
13. Li, M., Liu, S., Zhou, H., Zhu, G., Cheng, N., Deng, T., Wang, H. In: SGS-SLAM: Semantic Gaussian Splatting for Neural Dense SLAM. Springer Nature Switzerland (10 2024) 163–179
14. Chen, T., Shorinwa, O., Bruno, J., Yu, J., Zeng, W., Nagami, K., Dames, P., Schwager, M.: Splat-nav: Safe real-time robot navigation in gaussian splatting maps (2024)
15. Lei, X., Wang, M., Zhou, W., Li, H.: Gaussnav: Gaussian splatting for visual navigation (2024)
16. Jin, R., Gao, Y., Wang, Y., Lu, H., Gao, F.: Gs-planner: A gaussian-splatting-based planning framework for active high-fidelity reconstruction (2024)
17. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2) (1968) 100–107
18. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts (1996)
19. Şucan, I.A., Moll, M., Kavraki, L.E.: The Open Motion Planning Library. IEEE Robotics & Automation Magazine **19**(4) (December 2012) 72–82 https://ompl.kavrakilab.org.
20. Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., Kanazawa, A.: Nerfstudio: A modular framework for neural radiance field development. In: ACM SIGGRAPH 2023 Conference Proceedings. SIGGRAPH '23 (2023)
21. Ladd, A., Kavraki, L.: Motion planning in the presence of drift, underactuation and discrete system changes. In: Robotics: Science and Systems I. (06 2005) 233–240
22. Gipson, B., Moll, M., Kavraki, L.E.: Resolution independent density estimation for motion planning in high-dimensional spaces. In: 2013 IEEE International Conference on Robotics and Automation. (2013) 2437–2443
23. Otte, M., Frazzoli, E. In: RRTX: Real-Time Motion Planning/Replanning for Environments with Unpredictable Obstacles. Springer International Publishing, Cham (2015) 461–478
24. Gammell, J.D., Barfoot, T.D., Srinivasa, S.S.: Batch informed trees (bit*): Informed asymptotically optimal anytime search. The International Journal of Robotics Research **39**(5) (2020) 543–567
25. Shi, J.C., Wang, M., Duan, H.B., Guan, S.H.: Language embedded 3d gaussians for open-vocabulary scene understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2024) 5333–5343