FOCI: Trajectory Optimization on Gaussian Splats

Mario Gomez Andreu*[1], Maximum Wilder-Smith*[1], Victor Klemm[1],
Vaishakh Patil[1], Jesus Tordesillas[2], Marco Hutter[1]

***Abstract*— 3D Gaussian Splatting (3DGS) has recently gained popularity as a faster alternative to Neural Radiance Fields (NeRFs) in 3D reconstruction and view synthesis methods. Leveraging the spatial information encoded in 3DGS, this work proposes FOCI (Field Overlap Collision Integral), an algorithm that is able to optimize trajectories directly on the Gaussians themselves. FOCI leverages a novel and interpretable collision formulation for 3DGS using the notion of the overlap integral between Gaussians. Contrary to other approaches, which represent the robot with conservative bounding boxes that underestimate the traversability of the environment, we propose to represent the environment *and* the robot as Gaussian Splats. This not only has desirable computational properties, but also allows for orientation-aware planning, allowing the robot to pass through very tight and narrow spaces. We extensively test our algorithm in both synthetic and real Gaussian Splats, showcasing that collision-free trajectories for the ANYmal legged robot that can be computed in a few seconds, even with hundreds of thousands of Gaussians making up the environment. The project page and code are available at https://rffr.leggedrobotics.com/works/foci**

I. INTRODUCTION

Trajectory planning is integral to autonomous mobile robotics to ensure guided and safe operation. However, in order to make an informed decision, these planning algorithms heavily depend on the underlying environment representations. Popular representations include occupancy grids, signed distance fields, 3D Meshes, and point clouds.

Recently, NeRFs [1] have been proposed as a novel neural representation of the environment. They can be created from simple monocular images and they encode the environment as a fully connected neural network, mapping position in space and viewing direction to occupancy and color. However, they suffer from having slow inference speeds because new views have to be created using a computationally expensive ray-casting procedure. More recently, 3DGS [2] has been proposed as a promising alternative to NeRFs. Instead of using a neural network to represent the radiance field, it is defined explicitly by a set of 3D Gaussian ellipsoids. Because of this, new views can be efficiently generated by projecting these ellipsoids into the view plane instead of sampling the implicit density of a NeRF along a ray. Not only does this accelerate novel view synthesis on learned scenes, but the rendering speeds allow for rapid training of environments in a matter of minutes.

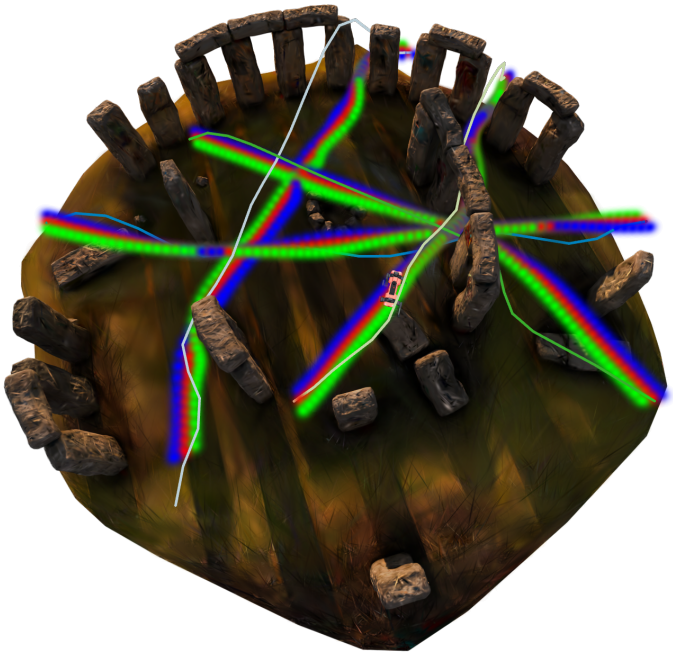


Fig. 1: Four different orientation-aware trajectories planned through a Stonehenge environment for an ANYmal legged robot. Splines show A* initial trajectories while green, red and blue Gaussians show an orientation-aware plan with green representing the front, red the center, and blue the rear.

Given the advantages that 3DGS offers when compared with NeRFs, the natural question is how a robot can leverage this Gaussian representation for navigation. **In this paper, we propose an algorithm that enables a robot to perform trajectory optimization directly on the 3D Gaussians.** Although some steps have been taken in this direction [3], [4], [5], the huge number of Gaussians that a scene can have, together with the specific formulation of an explicit collision measure, makes this problem especially hard.

To overcome these challenges, we propose **FOCI**, a trajectory optimization algorithm that leverages the overlap integral - the spatial integral over the multiplication of two functions - as a proxy measure for the collision between two Gaussians. By representing both the robot *and* the environment with 3D Gaussians, evaluating the full-body collision between them reduces to a sum of normal distribution evaluations. Furthermore, the resulting expression

*These authors contributed equally

[1]The authors are with the Robotic Systems Lab, ETH Zürich, Switzerland. {margomez, mwilder, vklemm, patilv, mahutter }@ethz.ch

[2]The author is with the IIT and ICAI, Comillas Pontifical University (Spain). jtordesillas@comillas.edu

This work was funded by NCCR Automation, and Swiss Federal Railways (SBB) via ETH Mobility Initiative.

	Splat-Nav [3]	GS-Planner [6]	GSM [5]	GaussNav [4]	FOCI (ours)
Planning pipeline included	✓	✓	✗	✓	✓
Orientation-aware collision	✗	✗	✗	✗	✓
Covariance information leveraged	✓	✓	✓	✗	✓

TABLE I: Comparison of our work with related works on Gaussian Splats

is fully differentiable, yielding expressive gradients in the optimization step.

The contributions of this work are therefore summarized as follows:

- A novel collision measure between Gaussian Splats based on the overlap integral between Gaussians.
- A trajectory planning algorithm that uses this formulation to optimize fully differentiable trajectories.
- A GPU implementation of the trajectory optimization.
- An evaluation of the proposed algorithm on the ANYmal legged robot in simulation and the real world.

II. Background and Related Work

Environmental representations are essential parts of the planning pipeline. They determine what methods can be used for trajectory generation, as well as what sensors and information are needed for real-world deployments.

A. Radiance Fields

Radiance Fields are widely used in visual computing to represent 3D scenes and perform novel view synthesis, creating photorealistic renders from out of domain poses. The preliminary form is a Neural Radiance Field, initially proposed by Mildenhall et al. [7], which are a type of environment representation that provide a neural mapping from position in space and a viewing angle to visual density and color along a ray. To render an image this neural radiance is integrated along a series of rays capturing view-dependent effects like specularity and volumetrics.

However, this ray casting operation can be costly at scale, and the implicit scene representation can make direct sampling difficult. As an alternative, 3DGS [2] produces similar quality reconstruction with an explicit basis at far faster speeds. Instead of encoding the scene in a neural network, 3DGS creates a series of 3D Gaussian ellipsoids with colors and Spherical Harmonics [8] to encode view-dependent effects. Instead of ray casting each Gaussian, the ellipsoids are splatted [9] onto the viewing plane for rasterization levels of speed. Similar to NeRFs, 3DGS is optimized by minimizing the deviation between an actual image and the corresponding image predicted by the model. Because of their explicit nature, 3DGS allows for an informed guess that initializes all Gaussian means to points from Structure from Motion. The photometric losses then optimize the Gaussians' colors, opacities and covariances to more accurately match the images. During training, adaptive density control will split, prune, and clone Gaussians ensuring there are just enough to faithfully represent the scene.

B. Trajectory Optimization

1) Standard Methods: Trajectory planning generates a path for a robot that is collision-free with the surrounding environment and further minimizes a performance metric such as execution time or control efforts. Depending on the formulation, trajectory optimization also includes optimizing control inputs to follow the trajectory. Similar to other works [3], [4], [6], we assume robust low-level control that can track reasonably smooth state trajectories, so we omit the control inputs in our optimization formulation.

Graph-based methods such as A* [10], and sampling-based methods such as RRT* [11] are simple to implement and produce a series of collision-free waypoints. Trajectories produced by graph-based methods are often good initial guesses for other approaches. Unlike graph- or sampling-based methods, spline-based methods optimize a continuous trajectory instead of a series of waypoints. Collision avoidance is ensured by either constraining the convex hull of this spline to a predetermined safety corridor or by ensuring no collision at discrete sample points along the spline. Zhou et al. [12] obtain an obstacle gradient to push spline trajectories out of obstacles to achieve collision-free trajectories while Gao et al. [13], [14] construct a safe flight corridor around an RRT* path through a point cloud and optimize a spline trajectory inside the corridor to ensure safety.

2) Radiance Field Methods: There are numerous trajectory optimizing algorithms designed to work on standard scene representation such as meshes or point clouds– and now with the recent emergence of Radiance Fields– NeRFs are being used to tackle the problem. Planning directly on Radiance Fields can be favorable as they are often trained from monocular images, and the extensive reconstruction allows for their use in multiple downstream tasks from planning and SLAM to teleoperation. Pantic et al. [15] propose a method to obtain an Euclidean Signed Distance Field (ESDF) directly from a NeRF by inserting an additional head into the neural network to output the encoded distance. Adamkiewicz et al. [16] directly use the opacity output by the NeRF as a collision measure at a set of points of the robot body. Starting from an A*-initial guess, a joint cost function consisting of a control effort and opacity term at the control points is minimized. In CATNIPS, [17] a probabilistic approach is used and shows that a NeRF is equivalent to a Poisson Point Process, which allows for direct quantification of the collision probability. Using this property, they formulate a chance constraint trajectory optimization.

Due to its novelty, few papers have been published on trajectory optimization for Gaussian Splats. In GaussNav [4], Lei et al. reduce the Gaussian Splat to a point cloud of

its means, which is voxelized and then projected to a two-dimensional cost map where trajectories are planned. This produces quick results but does not fully leverage the explicit 3D nature of 3DGS. SplatNav [3] additionally takes the covariances of the Gaussians into account and computes a polyhedral safety corridor around an A* initial guess. The free space is determined by calculating the distance between the robot ellipsoid and the confidence intervals of the environment Gaussians. Inside this safety corridor, a spline trajectory is optimized to produce a safe path. The GS-Planner [6] constrains the position along the trajectory to be outside of the 3σ confidence interval of the joint Gaussian of the ellipsoid and robot radius. Unlike SplatNav [3], which uses a safety corridor as an intermediate representation, this constraint is directly imposed on the spline. Goel and Tabib [5] propose a method to derive an ESDF and collision probability of a Gaussian Surface Model for an ellipsoidal robot with a fixed orientation, which can then be used to plan trajectories.

While these works present considerable steps towards 3DGS planning, they consider only ellipsoidal robots with a fixed orientation. However, in real world settings [18], [19], optimizing orientation is crucial to navigating through narrow passageways. **By leveraging a detailed whole-body overlap model, we propose a solution to this shortcoming**. Table I provides an overview of how our algorithm compares to related work.

III. Method

Our methodology can be split into three parts: *1)* trajectory representation to create an initial spline, *2)* collision measure and *3)* optimization loop.

A. Trajectory Representation

We employ cubic B-splines [20] for the optimization to generate a fully differentiable and smooth trajectories in d dimensions. With B-splines, every point along the trajectory can be expressed as a weighted sum of the H control points $\mathbf{Q}_i$ of the trajectory. The progress along the trajectory is usually parameterized by a progress variable s, which can (but is not required to) be equal to the system time t. Cubic B-splines have $H-4$ knots, which we set equidistantly to $\{0, 1, \dots, H-5\}$. Each point along the trajectory depends on four successive control points according to Equation 1 with $s' = s - i$.

$$\mathbf{x}(s') = \underbrace{\begin{bmatrix}1 & s' & s'^2 & s'^3\end{bmatrix} \frac{1}{6} \begin{bmatrix}1 & 4 & 1 & 0\\ -3 & 0 & 3 & 0\\ 3 & -6 & 3 & 0\\ -1 & 3 & -3 & 1\end{bmatrix}}_{\Phi} \begin{bmatrix}\mathbf{q}_i^T\\ \mathbf{q}_{i+1}^T\\ \mathbf{q}_{i+2}^T\\ \mathbf{q}_{i+3}^T\end{bmatrix} \tag{1}$$

By discretizing the trajectory with Δs into K intervals and constructing a matrix $\underline{\Phi}$ from Equation 1 evaluated at each progress step, the relationship between sample point matrix $\underline{\mathbf{x}} = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_K]^T \in \mathbb{R}^{K\times \mathrm{d}}$ and control point matrix $\underline{\mathbf{Q}} = [\mathbf{q}_1, \mathbf{q}_1, \dots, \mathbf{q}_H]^T \in \mathbb{R}^{H\times \mathrm{d}}$ is then given as

$$\underline{\mathbf{x}} = \underline{\Phi}\underline{\mathbf{Q}} \tag{2}$$

This parametrization works well for simple vector spaces such as $\mathbb{R}^3$ but fails to interpolate between orientations. Spline representations for Lie Groups have been discussed in robotics literature [21], which allow for pose optimization in SE(3), encoding orientations.

Although this formulation is fully compatible with the proposed framework, we decided to follow a simpler parameterization because our target platform is a legged robot. Instead of the full pose, **only the position $\mathbf{p}$ and yaw angle ψ are parameterized** to optimize control points $\mathbf{q}_i \in \mathbb{R}^4$.

B. Collision Measure

To quantify the collision between two objects, we want to measure the overlap between them. The collision measure should, therefore, fulfill the following three properties: *1)* It should be close to zero if two objects do not overlap. *2)* It should be greater than zero when two objects overlap and quantify the magnitude of that overlap. *3)* It should be differentiable to allow for gradient evaluations in optimization.

In our work, we model both the environment $p(\mathbf{x})$ and the robot $r(\mathbf{x})$ by 3D Gaussians, allowing us to define the density of the robot and the environment at a specific point in space as

$$p(\mathbf{x}) = \sum_{i=1}^{N} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{3}$$

$$r(\mathbf{x}) = \sum_{j=1}^{M} \mathcal{N}(\mathbf{x}; \bar{\boldsymbol{\mu}}_i, \bar{\boldsymbol{\Sigma}}_j) \tag{4}$$

The volume in which both quantities overlap is defined as the collision volume. To compute the magnitude of the collision, we compute the overlap integral between the environment field $p(\mathbf{x})$ and the robot field $r(\mathbf{x})$:

$$\int_{\mathbb{R}^3} p(\mathbf{x}) r(\mathbf{x}) dV = \sum_{i=1}^{N}\sum_{j=1}^{M} \int_{\mathbb{R}^3} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\mathcal{N}(\mathbf{x}; \bar{\boldsymbol{\mu}}_j, \bar{\boldsymbol{\Sigma}}_j) dV \tag{5}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{M} \mathcal{N}(\bar{\boldsymbol{\mu}}_j; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i + \bar{\boldsymbol{\Sigma}}_j) \tag{6}$$

Resulting in a representation that is fully differentiable with respect to the means $\bar{\boldsymbol{\mu}}_j$ and covariances $\bar{\boldsymbol{\Sigma}}_j$. Note that these robot means and covariances can further be parameterized by, for example, the base position and joint angles, whose gradients can then be derived with the chain rule.

This measure for collision fulfills the desired properties defined above. Because of Gaussian's exponential nature, it exponentially converges to zero as the distance between the robot and the obstacle increase. That also means it does not contribute meaningfully to the gradient at further distances.

Note that the quantity $\mathcal{N}(\bar{\boldsymbol{\mu}}_j; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i + \bar{\boldsymbol{\Sigma}}_j)$ can be interpreted as the exponential of the negative Mahalanobis distance between $\bar{\boldsymbol{\mu}}_j$ and $\boldsymbol{\mu}_i$ using the joint covariance of the robot and environment Gaussian.

$$\mathcal{N} = \exp(-\frac{1}{2}(\bar{\boldsymbol{\mu}}_j - \boldsymbol{\mu}_i)^T(\boldsymbol{\Sigma}_i + \bar{\boldsymbol{\Sigma}}_j)^{-1}(\bar{\boldsymbol{\mu}}_j - \boldsymbol{\mu}_i)) \tag{7}$$

$$= \exp(-\frac{1}{2}d_M(\bar{\boldsymbol{\mu}}_j, \boldsymbol{\mu}_i)) \tag{8}$$

The Mahalanobis distance takes the potentially asymmetric extent of the Gaussians into account, and the exponential ensures the desired decaying property of the measure.

C. Optimization Formulation

Since we use splines to represent the trajectory, we optimize the control points $\mathbf{q}_i = [\mathbf{p}_i^T, \psi_i]^T \in \mathbb{R}^4$ of the spline as decision variables. Due to the non-linear and constrained nature of the optimization problem, we employ the interior point method. Whenever integrating a quantity like the collision measure along the spline is necessary, we approximate it as a discrete sum over the values sampled along the spline in K equidistant steps. We represented the robot position and yaw orientation as a cubic B-spline. The kinematics of each Gaussian that makes up the robot is a function of the position and orientation of the base $\bar{\boldsymbol{\mu}}_j(\mathbf{p}, \psi)$.

We minimize the weighted sum of the obstacle cost, the jerk along the trajectory, and the distance of the final point to the goal with weights $\omega_1 = 0.1, \omega_2 = 40$ and $\omega_3 = 1$. The collision avoidance is included in the objective function instead of the constraints because the collision measure is not normalized, making finding an appropriate threshold unfeasible. A similar approach has been taken by [16], which faced a similar issue for NeRFs. We constrain the initial position of the trajectory to the current position $\mathbf{x}_0$. When planning for ANYmal, we further constrain the trajectory height to be at an appropriate distance h above the ground so that the robot can follow it.

We ensure that the velocity and acceleration remain within the physical limits of mobile system. Instead of directly constraining samples of the respective derivatives of the splat along the trajectory, we leverage the convex hull property of splines. Since each spline segment lies within the convex hull of its control points, it is enough to constrain the norm of the velocity and acceleration control points, to guarantee constraint satisfaction along the entire spline. However, the conservativeness of this over approximation depends on the choice of spline basis. Because of this, we compute the convex hull with respect the **MINVO Basis** [22], which results in convex hulls of minimal volume, and therefore the least conservative approximation.

The optimization problem is defined in Equation 9. Note that the spline is parameterized as a function of the progress variable s, we compute the time derivative with $\frac{d^i p}{dt^i} = \frac{d^i p}{ds^i}\left(\frac{ds}{dt}\right)^i$ and assume $m = \left(\frac{ds}{dt}\right)$ to be constant.

$$\min_{\mathbf{Q}} \ \omega_1 \sum_{k=0}^{K}\sum_{i=1}^{N}\sum_{j=1}^{M} \mathcal{N}(\bar{\boldsymbol{\mu}}_j(\mathbf{x}(k\Delta s), \psi(k\Delta s)); \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i + \bar{\boldsymbol{\Sigma}}_j) \tag{9}$$

$$+ \omega_2 \sum_{k=0}^{K} \| \ddot{\mathbf{x}}(k\Delta s)\| + \omega_3 \|\mathbf{x}(K\Delta s) - \mathbf{x}_{\text{goal}}\| \tag{10}$$

$$\text{s.t } \|\mathbf{x}(0) - \mathbf{x}_0\| = 0 \tag{11}$$

$$\|\mathbf{x}(k\Delta s)(2) - h\| = 0 \ \forall k \tag{12}$$

$$\|\dot{\mathbf{p}}(s)\| \leq v_{\text{max}} \forall s \tag{13}$$

$$\|\ddot{\mathbf{p}}(s)\| \leq a_{\text{max}} \forall s \tag{14}$$

$$\left|\dot{\psi}(s)\right| \leq \omega_{\text{max}} \forall s \tag{15}$$

$$\left|\ddot{\psi}(s)\right| \leq \alpha_{\text{max}} \forall s \tag{16}$$

In order to create an initial guess, the 3DGS is used to create a rough occupancy grid, where a voxel is occupied if it contains at least one mean. A* is then run on this occupancy grid and used to initialize the control points by fitting a spline to the resulting path.

D. Implementation Details

We define and solve the described optimization problem with Casadi [23]. We exploit the independent summation structure of the collision measure (Equation 5) and run this computation on the GPU in parallel. This led us to define a custom Casadi functor that computes the evaluation of the function and is gradients in parallel using NVIDIA Warp [24]. Note that other GPU extensions for Casadi, such as L4Casadi [25] and CusADI [26] exist, but were not directly applicable due to the custom 3D Gaussian overlap integral formulation. The optimization problem is then solved via the interior point method (IPOPT) [27] with the custom overlap integral functor.

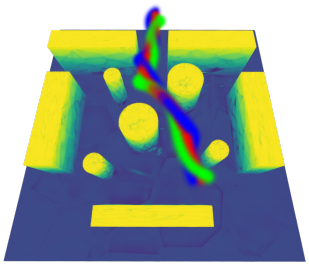
IV. Experiments

In this section, we evaluate our algorithm by applying it to planning problems in different environments represented by 3DGS. All experiments are run on an Intel i7-8750H with 16 GB of RAM and an NVIDIA RTX 2070 Max-Q.

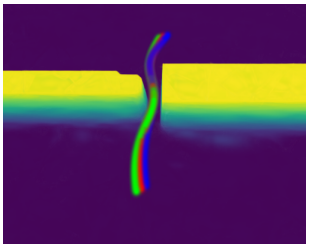
A. Trajectory Evaluation

To evaluate the functionality of our system, we plan trajectories through a range of 3DGS environments, both artificially generated scenes and ones created from real-world captures. Furthermore, we demonstrate the use of the algorithm for navigating an ANYmal robot on hardware to highlight the importance of orientation-aware planning.

1) ANYmal Navigation: In order to plan through a 3DGS scene, the robot must be properly localized in a correctly scaled Gaussian environment. In deployments, this was accomplished by performing ICP to align accumulated LiDAR data with the surface means of the Gaussian Splat. In simulation, synthetic scenes can be created and arbitrarily scaled to provide complex testing scenes.



(a) Pillar Environment



(b) Narrow Corridor

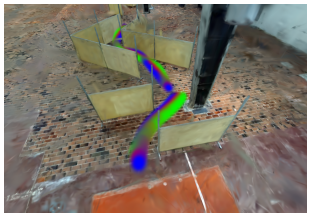
Fig. 2: Sample trajectories created on synthetic testing data showing a 3 Gaussian robot rotating to navigate the environments. The 3 Gaussians are shown in green for the front, red for the middle and blue for the rear of the robot, showing the rotation to navigate obstacles.

Environment	Initial Guess Time	Solve Time	# of Gaussians
Narrow Corridor	0.24s	0.47s	24k
Pillars	0.25s	0.45s	49k
Machine Hall	0.22s	2.12s	243k
Stonehenge	0.55s	0.83s	138k

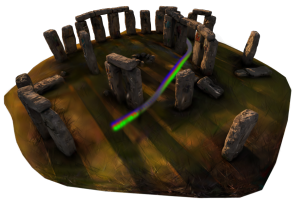
TABLE II: Planning time for synthetic 3DGS scenes (top) and realistic scenes (bottom) using a 3 Gaussian robot.

Figure 2a shows longer trajectories through the more complex pillar environment. The algorithm gracefully handles trajectories requiring additional turns while adapting the orientation to keep a maximum distance from the wall.

As Figure 2b shows, the planning algorithm effectively leverages the asymmetry of ANYmal to pass through the narrow opening collision-free. The start and end positions are constrained to be parallel to the wall; requiring the orientation to change to pass through the opening.



(a) Machine Hall



(b) Stonehenge Environment

Fig. 3: Trajectories planned for a 3 Gaussian robot rotating through realistic scenes.

2) General Trajectory Planning Through 3DGS: Figure 3 shows that we can plan collision-free trajectories through splats that were created directly from the real-world environments. In the case of the machine hall sample, the data used for the reconstruction came entirely from the on-board sensors of the robot [28], quickly providing a scene with accurate scaling.

3) Hardware Experiments: Finally, we showcase the method deployed on hardware using a real 3DGS reconstruction of a scene captured using drone imagery and localized on site using ICP on the LiDAR data and Gaussian means. The narrow passages visible in Figure 4 required the robot to rotate in order to safely traverse the environment, while obstacles on the ground forced the height-constrained trajectories to avoid shortcuts.

Fig. 4: Trajectories planned for the ANYmal robot through real world obstacles. On the bottom the actual robot can be seen at the site following the trajectories generated by our algorithm **FOCI**.

Method	Solve Time (s)	Path Length (-)	Min. Safety Dist. (m)
RRT*	100.06	0.92	0.51
Splat-Nav	**0.76**	1.01	0.56
Ours	0.78	**0.88**	**0.27**

TABLE III: Comparisons of the performance of our method with a similar 3DGS based planner and simple RRT* planner. Three metrics are used for evaluation, the speed of optimization, a path length relative to the scene scale, and minimum distance between the robot model and environment to measure the needed safety corridor.

B. Runtime

We evaluate the performance of our method by comparing the runtimes of the Casadi optimization on a single CPU core, multiple CPU cores, and the GPU. The computation time is generally linear with the number of environmental Gaussians, robot Gaussians, and collision sample points.

While a significant speedup can be observed when optimizing on multiple CPU cores using OpenMP, **our custom implementation is** 320 **times faster**, often resulting in solutions that only take a few seconds.

Table II show the total planning time for different environments of varying complexity. In the realistic environments, the larger scenes resulted in a longer A* search time, while the more complex 3 Gaussian robot slightly increases the

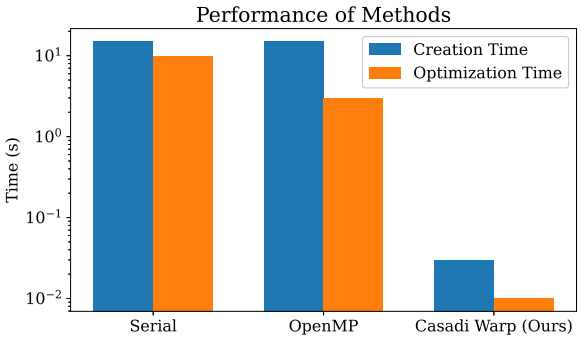


Fig. 5: Comparison of the solver's creation and runtime running on the CPU and GPU for 50k environmental Gaussians and one robot Gaussian. The "serial" method is on a single CPU core, "OpenMP" runs on multiple CPU cores and CasADi Warp is our custom GPU implementation.

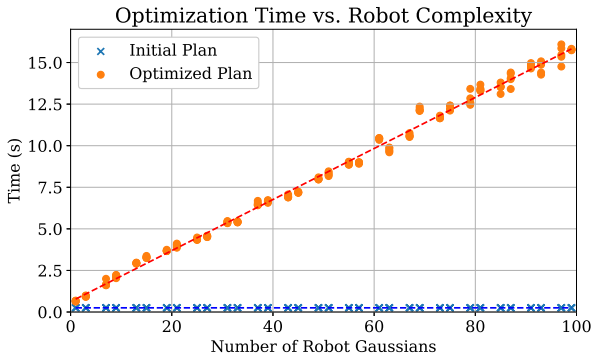


Fig. 6: Optimization time for orientation-aware planning using increasingly complex robot models. These are planned through the Stonehenge environment with 138k Gaussians.

general solve time. This relation between complexity of the robot model and solve time forms a linear relation as shown in Figure 6.

In comparisons with similar methods we are able to **surpass the speed of traditional methods such as RRT* on large complex scenes**, while having **similar time performance to state-of-the-art 3DGS methods such as Splat-Nav**. Additionally, our orientation-aware planning allows us to use halve the safety corridor distance in one axis to model a 1m by 0.5m robot such as ANYmal. Due to this unique ability to model the robot as a collection of Gaussians and therefore consider the robot's orientation, **the smaller safety corridor allowed for slightly shorter paths, leveraging the robot's geometry**.

In Figure 6 we compare the effect of more complex robot Gaussians on the solve time for large complex scenes such as Stonehenge. A **linear relation between robot complexity and optimization time** shows the potential of modeling more complex robot geometry, allowing even tighter safety corridors on more complex robots.

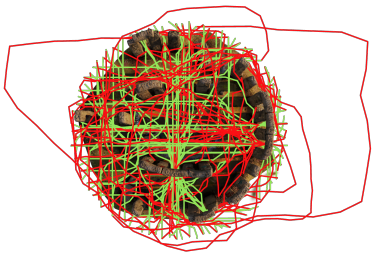


Fig. 7: The collection of paths used to evaluate different methods with various start and end goals around Stonehenge. The RRT* path is is shown in red and Splat-Nav[3] in green.

V. Limitations

Three current shortcomings of the algorithm include *a)* occasional obstacle collision, *b)* distance agnostic optimization, *c)* sensitivity to 3DGS quality.

a) Both the acceleration constraint as well as the obstacle cost are additive terms in the cost function. Since obstacle avoidance is not formulated as a hard constraint, it can be traded off with the acceleration cost, yielding a low acceleration but colliding trajectories. Figure 8 shows a trajectory resulting from such a trade off. Related work, such as trajectory planning on NeRFs by Adamkiewicz et al. [16], has also encoded obstacle avoidance as a cost function component. We are confident that the existing issues can be resolved by constraining the trajectory to be close to the initial guess, tuning the weights of the individual cost terms, or using higher-order derivatives of the trajectory as the effort cost.

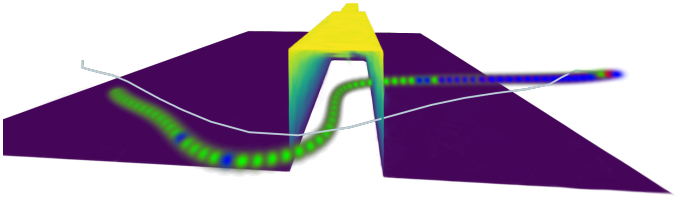


Fig. 8: Optimized trajectory for which the collision avoidance fails.

b) Trajectories are parameterized over an interval that does not depend on the physical duration of its execution but on the number of used control points. While this formulation is easy to implement, it has impactful disadvantages. Between two trajectories of the same shape but different lengths, the shorter one has a lower acceleration cost than the longer one because the robot has to be accelerated over larger distances while still being parametrized over the same progress interval. This means that increasing the distance between the start and end pose leads to the acceleration cost becoming the dominant term in the optimization, decreasing the importance of obstacle avoidance. These effects can be overcome by making the optimization distance-aware. The acceleration cost could be scaled by the length of the initial guess between the start and the goal. Alternatively, the execution time between the start and goal could be introduced into the

trajectory representation and directly optimized as a decision variable.

c) By using the overlap integral to compute collision between the robot and the scene, we assume that areas of high Gaussian overlap in the environment are dense objects geometrically, however this is not always the case. Instead, the 3DGS optimization process results in a high number of overlapping Gaussians in areas of high information density, both in terms of texture and geometric data. More complex shapes such as edges, fine strands, or lettering result in a large amount of Gaussians to accurately capture the geometry and texture. This means that when computing the overlap integral over the environment, flat regions with text or patterns have a slightly higher collision cost than clean flat surfaces. Additionally, as splatting only renders and optimizes Gaussians visible to the camera, the internal Gaussian density of objects is not guaranteed. This means that if the trajectory does collide with a wall it can get stuck in a local minima and not recover with the aid of internal collision gradients. Both of these can be addressed with more intelligent 3DGS creation, either by ensuring the Gaussians mainly represent geometry [29], or ensuring internal object density is consistent [5].

VI. Conclusion

In this work, we proposed a novel collision formulation for 3D Gaussian Splatting, that is computed in an efficient manner, and integrated it into a trajectory optimization pipeline. We show that it can be effectively used for orientation-aware planning and verify it on the ANYmal robotic platform. Because we exclusively operate in 3D Gaussian space, representing the environment and the robot as Gaussians, our method can be freely combined with new developments from the 3DGS community. Furthermore, we show this method works on realistic data including scenes captured using the onboard sensor of the robot itself.

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: representing scenes as neural radiance fields for view synthesis," *Commun. ACM*, vol. 65, no. 1, p. 99–106, Dec. 2021. [Online]. Available: https://doi.org/10.1145/3503250

[2] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, 8 2023.

[3] T. Chen, O. Shorinwa, W. Zeng, J. Bruno, P. Dames, and M. Schwager, "Splat-nav: Safe real-time robot navigation in gaussian splatting maps," 3 2024. [Online]. Available: http://arxiv.org/abs/2403.02751

[4] X. Lei, M. Wang, W. Zhou, and H. Li, "Gaussnav: Gaussian splatting for visual navigation," *arXiv preprint arXiv:2403.11625*, 2024.

[5] K. Goel and W. Tabib, "Distance and collision probability estimation from gaussian surface models," *arXiv preprint arXiv:2402.00186*, 2024.

[6] R. Jin, Y. Gao, Y. Wang, Y. Wu, H. Lu, C. Xu, and F. Gao, "Gs-planner: A gaussian-splatting-based planning framework for active high-fidelity reconstruction," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 11 202–11 209.

[7] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," 2020.

[8] Fridovich-Keil and Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *CVPR*, 2022.

[9] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "Ewa volume splatting," in *Proceedings Visualization, 2001. VIS '01.*, 2001, pp. 29–538.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[11] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1478–1483.

[12] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2021.

[13] F. Gao and S. Shen, "Online quadrotor trajectory generation and autonomous navigation on point clouds," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2016, pp. 139–146.

[14] F. Gao, W. Wu, W. Gao, and S. Shen, "Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments," *Journal of Field Robotics*, vol. 36, pp. 710–733, 6 2019.

[15] M. Pantic, C. Cadena, R. Siegwart, and L. Ott, "Sampling-free obstacle gradients and reactive planning in neural radiance fields," in *Workshop on" Motion Planning with Implicit Neural Representations of Geometry" at 2022 IEEE International Conference on Robotics and Automation (ICRA 2022)*, vol. 1, 2022.

[16] 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.

[17] T. Chen, P. Culbertson, and M. Schwager, "CATNIPS: Collision avoidance through neural implicit probabilistic scenes," *IEEE Transactions on Robotics*, 2024.

[18] X. Dong, Y. Cui, J. Xiang, D. Li, and Z. Tu, "An efficient trajectory generation for bi-copter flight in tight space," *arXiv preprint arXiv:2406.00671*, 2024.

[19] N. Funk, J. Tarrio, S. Papatheodorou, P. F. Alcantarilla, and S. Leutenegger, "Orientation-aware hierarchical, adaptive-resolution a* algorithm for uav trajectory planning," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6723–6730, 2023.

[20] F. Holmér, "The continuity of splines," https://www.youtube.com/watch?v=jvPPXbo87ds, December 2022, accessed: 2024-06-26.

[21] C. Sommer, V. Usenko, D. Schubert, N. Demmel, and D. Cremers, "Efficient derivative computation for cumulative b-splines on lie groups," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. [Online]. Available: http://dx.doi.org/10.1109/CVPR42600.2020.01116

[22] J. Tordesillas and J. P. How, "Minvo basis: Finding simplexes with minimum volume enclosing polynomial curves," *Computer-Aided Design*, vol. 151, p. 103341, 2022.

[23] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[24] M. Macklin, "Warp: A high-performance python framework for gpu simulation and graphics," https://github.com/nvidia/warp, March 2022, nVIDIA GPU Technology Conference (GTC).

[25] T. Salzmann, J. Arrizabalaga, J. Andersson, M. Pavone, and M. Ryll, "Learning for casadi: Data-driven models in numerical optimization," 2023.

[26] S. H. Jeon, S. Hong, H. J. Lee, C. Khazoom, and S. Kim, "Cusadi: A gpu parallelization framework for symbolic expressions and optimal control," *IEEE Robotics and Automation Letters*, 2024.

[27] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

[28] M. Wilder-Smith, V. Patil, and M. Hutter, "Radiance fields for robotic teleoperation," *arXiv*, 2024.

[29] B. Chao, H.-Y. Tseng, L. Porzi, C. Gao, T. Li, Q. Li, A. Saraf, J.-B. Huang, J. Kopf, G. Wetzstein, and C. Kim, "Textured gaussians for enhanced 3d scene appearance modeling," 2024. [Online]. Available: https://arxiv.org/abs/2411.18625