Hallucinating The Best Shot: Towards Geometrically-Aware Generative Cinematography

Timothy Chen[1]

***Abstract*—We present Cine-NeRF, a generative cinematography pipeline that automatically plans photo-realistic camera paths to generate novel video streams to inform cinematographers of how to get the best shot. The pipeline relies on a 3D reconstruction of the environment from an input image stream, namely a Gaussian Splatting (GSplat) environment representation. The pipeline is user-friendly, in that a cinematographer provides textual queries of which objects for the camera to go to and which objects for the camera to look at. To encourage clear shots of the objects of interest, we formulate rigorous collision avoidance constraints of the camera with the Gaussian Splat that are computationally efficient. We then build guaranteed-safe polytope corridors through the scene and optimize the camera trajectory through this corridor. The user can reinitialize the pipeline with random initial starting points to generate novel, photo-realistic, and dynamically feasible video streams so that cinematographers can choose the best camera trajectory to take videos from in the real world.**

I. Introduction

Cinematographers constantly face the labor-intensive challenge of chasing the best shot. The perfect shot may necessitate many trials of slightly different camera trajectories and scene conditions. This necessitate the gathering of new videos for each trial, even though the perfect shot may be a small fraction of the total frames collected. Not only is this time inefficient, but it is also memory inefficient, as cinematographers need to store past video streams for reference. We believe that building a tool (i.e. a photo-realistic "simulator") where users can hallucinate the entirety of the environment from just one set of video and autonomously generate photo-realistic and physically plausible video streams will facilitate the cinematographic process.

The creation of a simulator requires a 3D representation of the scene. Constructing a representation of the environment from images such as RGB cameras is a fundamental challenge in research. Traditionally, these representations have included (watertight) triangular meshes [6], occupancy grids [7], point clouds [13], and Signed Distance Fields (SDFs) [21]. In each of these situations, the geometry is well-defined and there are many methods for camera planning. However, these tools may be computationally expensive to compute, lack photo-realism, and lack generalizability to fit into modern machine learning pipelines such as learning semantics of the scene.

Neural Radiance Fields (NeRFs) [18] is a promising 3D scene representation alternative. NeRFs offer several potential benefits over traditional scene representations: they provide photo-realistic renderings, they can be trained using only monocular RGB images, and current implementations can train NeRFs in seconds [20, 24].

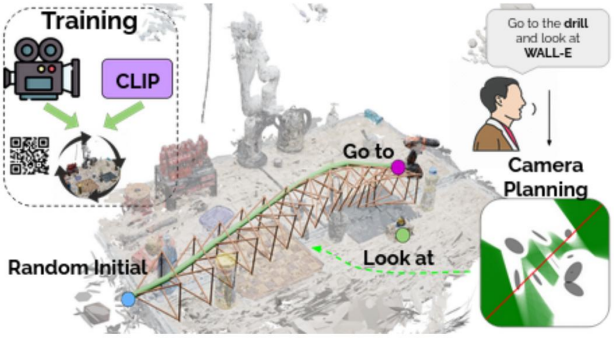

Fig. 1. Our real-time video generation pipeline, Cine-NeRF, consists of an implementation of semantic embedding into Gaussian Splats and a fast, physically plausible camera planning scheme. We encode pixel-aligned CLIP features into a latent space, which we distill into the per-Gaussian semantic attribute along with the standard GSplat attributes. This allows users to textually identify objects of interest in the scene. For camera planning, we first represent the camera as a ball, and then rapidly solve for smooth paths using an ellipsoid-to-ellipsoid collision test. The camera paths are biased towards the textually-identified objects of interests. Our code can be found on github.

More recently, Gaussian Splatting (GSplat) [11] has emerged as an extension of NeRF. GSplat represents the environment as a union of Gaussian/ellipsoidal primitives, learning the number of primitives along with the position, extents, color, and opacity of each. This approach allows for faster reconstruction by taking advantage of fast differential rasterization techniques for 3D Gaussians on the GPU and often yields more photo-realistic renderings compared to NeRFs. More importantly for tasks that interact with the geometry, the use of (convex) geometric primitives facilitates the development of interpretable and efficient planning pipelines [4].

In this paper, we develop a user-friendly cinematographic framework called Cine-NeRF, outlined in Fig. 1, for camera planning and video generation that is intertwined with the GSplat representation. Our overarching goal is to ingest a single video stream to build a GSplat, which can then be used to generate photo-realistic video streams from random trajectories. Our first contribution is the implementation of an augmented GSplat representation with semantic embeddings in order to allow for textual instructions from the user to bias the camera paths. Namely, the user can specify where the camera should *go to* and where it should *point at*. For

[1]Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA chengine@stanford.edu

example, the user can type "Go to the palm tree and look at the umbrella". The second contribution is the rigorous quantification of collision avoidance in GSplats, applied to camera trajectories. This is necessary in order to generate physically plausible trajectories, as ones that go through objects are not only visually unpleasant, but impossible to reproduce in the real world, and thus a waste of computational resources. As a result, we develop a computationally efficient collision checker that is guaranteed to be safe. Using this collision checker, we can parametrize the camera trajectories as smooth splines that are guaranteed to be a physically feasible shot. These modules are many times faster than their existing neural network-based NeRF counterparts, which do not provide guarantees on performance [1].

To showcase these contributions, we provide qualitative and quantitative results to show that our pipeline: (1) generates collision-free and smooth trajectories through the environment, (2) generates a photo-realistic image stream, and (3) can be executed in real-time. We demonstrate that we can generate new camera trajectories at a rate of 5 Hz on a desktop computer, facilitating rapid iteration toward finding the best shot.

II. Related Work

A. Traditional Camera Planning

There is a long history of virtual camera planning, which is encapsulated in a survey by Christie [5]. The survey classifies four approaches to the problem: algebraic, interactive, real-time reactive, and optimization-based. Our method does not fall into any single category, but blends interactivity with real-time automation in an optimization framework in order to generate the desired camera trajectories. There does exist other extensive pipelines that include 3D reconstruction, artistic shot selection, and motion planning [2], but relies on access to a depth camera. However, in this work, we only focus on the most basic of necessities, that is collision-avoidance of the camera. We do not reason about occlusions, lighting, or other artistic properties in this work.

For more general purpose planning, LaValle [14] provides an excellent explanation of the major algorithms in this realm. Most relevant for this work are the graph-based planners (e.g., A*), which compute a path over a grid representation of the environment; sampling-based planners (e.g., PRM [10], RRT [15], and RRT* [9]), which generate a path by sampling candidate states within the configuration space. These methods will be useful later, as we can use them for path initialization.

B. NeRF Planning

NeRFs implicitly represent the environment as a spatial density field (with color) [18]. There are a number of works on planning in this representation. Adamkiewicz et al. [1] plan trajectories for differentially flat robots, such as quadrotors, that minimize a total collision cost. Chen et al. [3] convert the NeRF into a probabilistic voxel grid and then use this to generate trajectories parameterized as Bézier curves.

C. Gaussian Splats and Semantics

To the best of our knowledge, there are no planning algorithms designed to operate in GSplat environments except for [4]. There also exists several semantic extensions to Gaussian Splatting, but our implementation is most inspired by [22], which learns an encoder-decoder structure for the semantic channel. There are also a few recent works on SLAM using a GSplat representation of the environment [25, 26, 17] should we extend the work to include real-time scene building.

III. Semantic Modelling

A. Training and Representation

For brevity, we will not discuss the details of the basic Gaussian Splatting representation and training pipeline, but reference readers toward [11] for an in-depth discussion. Instead, we will detail the semantic modelling.

Our approach leverages the ellipsoidal representation of objects in a GSplat. An ellipsoid $\mathcal{E}_j$ is parameterized by covariance Σ_j and mean μ_j:

$$\mathcal{E}_j = \{x \in \mathbb{R}^3 \mid (x-\mu_j)^T \Sigma_j^{-1} (x-\mu_j) \leq 1\}, \tag{1}$$

where the covariance can be represented as a scaling matrix and a rotation matrix in its eigen-decomposed form.

In addition, each Gaussian also has an opacity, representing how intense the light transmitted is, and an RGB color. In our semantic model, we additionally learn a latent encoding of length 3, an encoding of the full semantic embedding. This compression is necessary for computational speed and memory constraints. In order to learn this additional encoding, we first learn an encoder-decoder that maps vectors of length 1024 representing Contrastive Language Image Pretraining embeddings (CLIP) to themselves and is separate from the GSplat training pipeline. The encoder and decoder are each instances of a fully-fused multi-layer perceptron [19]. Specifically, we learn the mapping

$$\hat{P} = f_{dec}(f_{enc}(P)). \tag{2}$$

After the GSplat is trained, we then instantiate the distillation of the latent encoding into the Gaussians. Namely, we render the latent image using the frozen opacities

$$\hat{l} = \sum_i^N l_i \prod_j^i (1-\alpha_j)\alpha_i \tag{3}$$

and regress l_i using a L2 loss between $\hat{l}$ and $f_{enc}(P)$ and performing gradient descent. The full model is visualized in Figure 2.

B. Inference

The semantic embedding of the GSplat allows for free-text queries to the model and associations of this query to regions of 3D space. To generate a metric for this association (i.e. relevancy), we rely on the cosine similarity between the decoded Gaussian CLIP codes and the full-dimensional CLIP code of the text query. To create a pseudo-probability that a

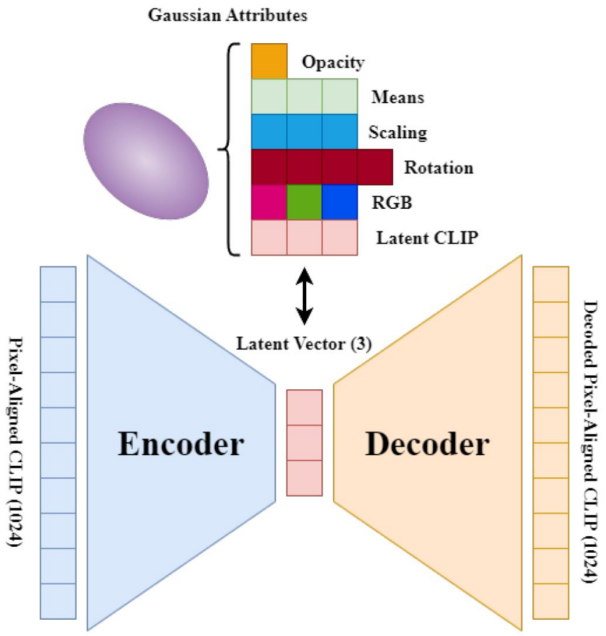

Fig. 2. A single Gaussian in a Gaussian Splat has attributes of opacity, color, the parameters of the ellipsoid, as well as a compressed CLIP code. The code is distilled from a learned encoder-decoder of the full-dimensional CLIP codes.

particular Gaussian is associated with the text query, we use the cosine similarity in a softmax across "negative" queries such as "stuff", "things", and "texture", as was done in [12]. Specifically, the relevancy is computed as

$$\min_i \frac{\exp(\phi_{query} \cdot f_{dec}(l_{gauss}))}{\exp(\phi_{query} \cdot f_{dec}(l_{gauss})) + \exp(\phi^i_{negative} \cdot f_{dec}(l_{gauss}))}, \tag{4}$$

where ϕ_{query} is the CLIP embedding of the text query, l_{gauss} is the 3-dimensional compressed CLIP code of the Gaussian, and $\phi^i_{negative}$ are the CLIP embeddings for the "negative" queries. We consider an object as the union of Gaussians that have a relevancy above some threshold. In Figure 1, the drill and WALL-E are lit up in the scene, reflecting the Gaussians that correspond with the provided text queries.

IV. Planning

Now we present our camera planning pipeline based on Gaussian Splatting. The pipeline consists of generating safe polytopic corridors (inspired by [16]) that discretizes the free space of a GSplat. The method is real-time and produces smooth paths to reduce jerky motions.

Given a bounding ellipsoid $\mathcal{E}_\mathcal{R}$ for the immediate camera volume and a union of N ellipsoids $\mathcal{G} = \{\mathcal{E}_j\}_{j=1}^N$ (from GSplat), we seek to find a feasible path from an initial position X to a goal position Y such that there are no collisions, i.e., $\mathcal{E}_\mathcal{R} \cap \mathcal{E}_j = \emptyset$, $\forall \mathcal{E}_j \in \mathcal{G}$. Because we would like random camera trajectories produced by the pipeline, the initial position is random. The goal point is determined by the centroid of the object associated with the "go to" text query. Meanwhile, the orientation is always fixed to the centroid of the object associated with the "look at" query.

A. Collision Detection

The environment and the camera volume are represented completely by ellipsoids or unions thereof. Therefore, collision is identical to finding intersection between two ellipsoids (the camera and an ellipsoid $\mathcal{E}_j$ from the environment). This is given by the following Theorem.

Theorem 1. *Consider two ellipsoid $\mathcal{E}_a, \mathcal{E}_b$ (with means μ_a, μ_b and covariances Σ_a, Σ_b) and the concave scalar function $K : (0,1) \to \mathbb{R}$,*

$$K(s) = (\mu_b - \mu_a)^T \left[\frac{1}{1-s}\Sigma_a + \frac{1}{s}\Sigma_b\right]^{-1} (\mu_b - \mu_a).$$

Then, $\mathcal{E}_a \cap \mathcal{E}_b = \emptyset$ if and only if there exists $s \in (0,1)$ such that $K(s) > 1$.

The proof for this is stated in Proposition 2 in Gilitschenski and Hanebeck [8]. Note that while K is concave in s (i.e., $-K$ is convex in s), it is *not* concave with respect to the means and variances.

Theorem 1 is a complete test that will always indicate whether two ellipsoids are in collision or not, however it is not computationally easy to solve. The authors of [4] avoid the computation of the matrix inversion by extracting the rotations and scalings of the Gaussian Splat and parametrizing the camera body as a sphere.

Corollary 1. *Gaussian Splatting parametrizes the covariance of the jth ellipsoid as $\Sigma_j = RSS^TR^T$, where R^T is an orthonormal matrix that rotates the world coordinates into a body aligned frame, and S is a diagonal matrix representing the axes lengths. Let $SS^T = \mathbf{diag}(\lambda_i)$ and let $w = R^T(\mu_\mathcal{R} - \mu_j)$. In addition, if we choose to parameterize our robot body as a sphere with covariance $\Sigma_\mathcal{R} = \kappa\mathcal{I}$ then the intersection test can be simplified to*

$$K(s) = w^T \mathbf{diag}\left(\frac{s(1-s)}{\kappa + s(\lambda_i - \kappa)}\right) w.$$

Our contribution is the development of an alternative collision test derived from Proposition 1, and contrasts with Corollary 1 in that it can accommodate general ellipsoidal camera bodies and avoids having to perform an generalized eigendecomposition as is proposed by Proposition 1 of [4]. We present this result in the following proposition.

Proposition 1. *The concave function K in the intersection test given in Theorem 1 can be simplified to:*

$$K(s) \coloneqq (\mu_a - \mu_b)^T V_3(s)(\mu_a - \mu_b), \tag{5}$$

where $V_3 : \mathbb{R} \to \mathbb{R}^{3\times3}$ represents a matrix-valued function of s. For simplicity, we present $V_3(s)$ as a composition of the following functions of s:

$$
\begin{aligned}
V_0(s) &\coloneqq R_a \Lambda_a(s) R_a^T, \\
V_1(s) &\coloneqq V_0(s) - \frac{V_0(s) d_{b,(1)} r_{b,(1)} r_{b,(1)}^T V_0(s)}{1 + d_{b,(1)} r_{b,(1)}^T V_0(s) r_{b,(1)}}, \\
V_2(s) &\coloneqq V_1(s) - \frac{V_1(s) d_{b,(2)} r_{b,(2)} r_{b,(2)}^T V_1(s)}{1 + d_{b,(2)} r_{b,(2)}^T V_1(s) r_{b,(2)}}, \\
V_3(s) &\coloneqq V_2(s) - \frac{V_2(s) d_{b,(3)} r_{b,(3)} r_{b,(3)}^T V_2(s)}{1 + d_{b,(3)} r_{b,(3)}^T V_2(s) r_{b,(3)}},
\end{aligned}
$$

where $\Lambda_a : \mathbb{R} \to \mathbb{R}^{3\times3}$ yields a diagonal matrix, whose ith diagonal element is given by $\frac{(1-s)}{\lambda_{a,(i)}}$, $r_{b,(i)} \in \mathbb{R}^3$ denotes the ith column of R_b, and $d_{b,(i)} \coloneqq \frac{\lambda_{b,(i)}}{s}$.

Proof. The proof follows from application of the Sherman-Morrison formula [23]. □

Computing the maximum of K in (5) can still prove challenging. However, a sampling-based procedure might be more efficient, as noted in [4]. Specifically, we can compute the value of K over finite samples of s to determine the existence of an intersection between a pair of ellipsoids. This sampling-based procedure is guaranteed to be safe, albeit conservative, i.e., the sampling-based procedure will never identify a pair of intersecting ellipsoids as being intersection-free.

We do concede that the solving times of (5) are slower than those of Proposition 1 in [4]. The generalized eigen-decomposition only happens once for every camera-Gaussian pair (i.e. there is no dependency on s), and the contribution of s to the evaluation of K is solely in the instantiation of a single diagonal matrix for every s. Meanwhile, all steps in Proposition 1 from V_0 to V_3 depend on s, increasing the memory and computational costs even though it is a straightfoward sequence of matrix multiplications. However, this representation of K is more amenable for use in off-the-shelf optimization solvers and is used in the rest of the work.

B. Safe Polytopes

We emphasize that having convex primitives (ellipsoids) as an environment representation facilitates development of interpretable algorithms. The remaining pipeline of parametrizing the free space of Gaussian Splats as polytopes is identical to that proposed in [4]. Therefore, we do not provide further details except in the statement of the polytope creation. Proposition 2 of [4] can be directly adapted to work with Proposition 1.

Proposition 2. *Given a test point x^* (i.e., camera position $\mu_{\mathcal{R}}$) and a collision test set $\mathcal{G}^*$, for every ellipsoid in $\mathcal{G}^*$, a supporting hyperplane to the j-th ellipsoid (considering the radius around the camera) derived from Proposition 1 is given by*

$$
\underbrace{\Delta_j^T Q_j}_{a_j} x \geq \underbrace{\frac{k_j^*}{1+\epsilon} + \Delta_j^T Q_j \mu_j}_{b_j},
$$

for any $\epsilon > 0$ where $\Delta_j = x^* - \mu_j$, $s^* = \arg\max_{s\in(0,1)} K(s)$, $Q_j = \phi \ \mathbf{diag}\left(\frac{s^*(1-s^*)}{1+s^*(\lambda_i-1)}\right) \phi^T$, and $(k_j^*)^2 = K(s^*) = \Delta_j^T Q_j \Delta_j > 0$. Note that Q_j is always symmetric, positive-definite. By stacking the hyperplane constraints (a_j, b_j), we arrive at a polytope $Ax \geq b$ that is guaranteed to be safe.

Proof. The proof follows immediately from that of Proposition 2 in [4]. □

For completeness, we generate the same figure of the polytope representation of free space (Figure 2 of [4]) in Figure 3 using instead Proposition 2 and 1. The two figures are identical.

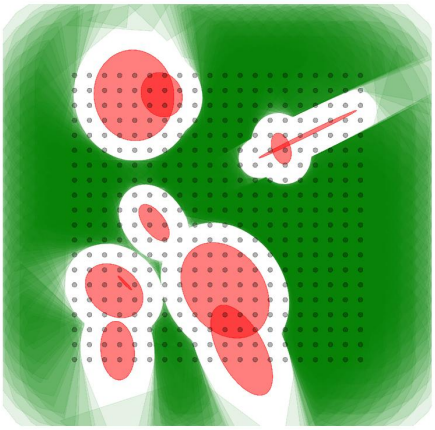

Fig. 3. Our supporting hyperplane algorithm finds safe polytopes (green) in the presence of ellipsoidal obstacles (red). Our algorithm is space-filling, shown by querying points on a grid (black dots) representing a camear with some radius. The resulting union of safe polytopes converges to the full extent of the collision-less space in the limit. It is also identical to Figure 2 of [4], validating completeness of Proposition 1.

C. Generating Paths

We again borrow much of the infrastructure for path generation from [4], so we avoid in-depth discussion. However, we summarize the process, as well as discuss how the objects of interest factor into planning.

1) Safe Camera Corridors: Splat-Nav [4] generates smooth, collision-free motion plans using the safe flight corridor method from [16], leveraging Proposition 2 to generate safe polytopes. The method uses A* to bias the final path around. Each point on the A* is queried through Proposition 2 to create a polytope, and the sequence of points correspond to a connected sequence of polytopes.

2) Camera Planning: To find a smooth, dynamically feasible trajectory, we parametrize the camera path as a sequence of N Bezier curves, one for each polytope created by the A*. Bezier curves have the convex hull property, which allow us to constraint the Bezier curves to lie within a particular polytope if we constraint the curve's control points to be within the same polytope. Therefore, we have a set of affine inequality constraints in the form of polytope constraints.

Additionally, we have equality constraints in the form of continuity and configuration. Continuity can be satisfied by making the last control point of a particular Bezier curve equal to the first control point of the next curve, an affine constraint. Similarly, we can make the first Bezier curve start at the randomly initialized point by setting the first control point of the first curve equal to that initial point. Finally, we do the same with the final control point of the final Bezier curve with respect to the goal location.

The goal location is the closed point in free space to the centroid of the "go to" object Gaussians. Much of the time, the centroid will be within the object and thus inaccessible. To project the centroid to the closest point in free space, we query Proposition 2 at the centroid, creating a polytope that is guaranteed safe and is the closest distance to the centroid. If no centroid can be created, we use the closest free space voxel that does not contain a Gaussian mean.

All the constraints are affine and can be embedded in a quadratic program, which can be solved efficiently. Specifically, we optimize over all control points of all Bezier curves such that the resulting path is of minimum length and hence smooth.

The camera trajectory thus far only spans positions, leaving the orientation as free variables. We can then always constrain the camera to orient itself toward the "look at" object's centroid x_{look} for all camera positions x_0. We first assume the X (right), Y (up), and Z (back) convention. We always point the -Z vector toward the "look at" point, therefore $\hat{Z} = -\frac{x_{look}-x_0}{||x_{look}-x_0||_2}$. Next, we want the Y to be oriented close to the Z axis of the world frame (i.e. aligned with gravity). Therefore, $\hat{X} = Z_{world} \times \hat{Z}$ and $\hat{Y} = \hat{Z} \times \hat{X}$ in a right-handed camera frame. The camera-to-world rotation matrix can be expressed as $R_{cam2world} = [\hat{X}|\hat{Y}|\hat{Z}]$.

V. Results

We examine the performance of our generative video pipeline on a real scene trained from real RGB images, shown in Fig. 4. First, we use a more probabilistic representation of the Gaussian Splats as proposed by [4], where we scale the GSplat covariances by a factor $\beta_j \alpha_j$, where β_j is related to the confidence interval of the Gaussian and α_j is the opacity of the Gaussian. This scaling represents the joint probability of an object existing in an ellipsoidal region of space and the "solidness" of that object. For these experiments, we take $\beta_j = 2$. For all tests, we parametrize the camera volume as a sphere with radius $r = 0.03$. The camera planning module runs at 5 Hz, while the video generation from rendering the NeRF can take a few seconds depending on the length of the trajectory. However, Gaussian Splatting is shown to be an order of magnitude faster to render than standard NeRFs [11]. All tests were performed on an RTX 4090.

Visually, the paths and image streams generated by Cine-NeRF are smooth, collision-free, and abide by their textual instructions in several settings (Fig. 4). This fact is validated in Fig. 5, where the camera trajectories are safe (distances greater than 0) with respect to the mesh of the Gaussian Splat. We also see that the semantic masks of the objects relevant to the text queries are also meaningful as they closely approximate the true object.

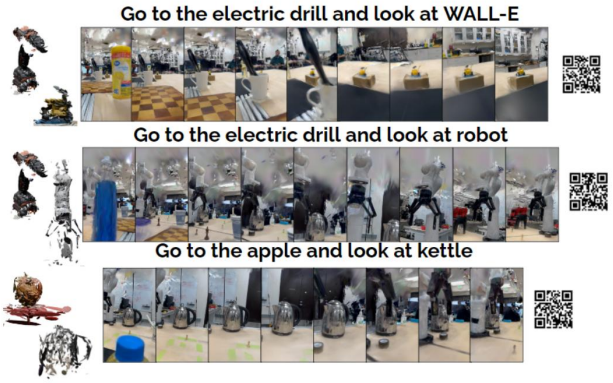

Fig. 4. Example frames corresponding with different textual queries of what to look at and go to. Visualizations of the semantic masks of objects are pictured to the left, and a QR code of the video is located on the right. Visually we see that (1) the camera is always oriented at the object of interest, (2) the camera goes toward the "go to" object, and (3) the object maintains a distance away from any objects in the scene. Additional videos can be found here.

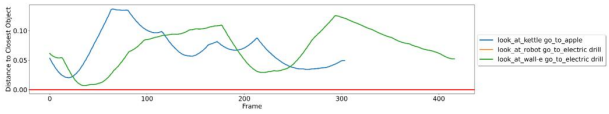

Fig. 5. Distances of the camera to the closest object along trajectories in Figure 4. The red line at 0 represents collision of any point of the camera ball with the scene. We see that the camera always maintains a positive distance away from obstacles, leading to a dynamically feasible, visually appealling shot.

VI. Conclusion

We introduce an efficient camera planning and visually appealing video generation pipeline termed Cine-NeRF, powered by a Gaussian Splatting 3D representation. The pipeline is user-friendly, where the only inputs to the pipeline is a single stream of images, and open-world text queries to bias the generated camera paths at test time. The camera planning pipeline is guaranteed-safe and real-time (5 Hz) by leveraging the ellipsoidal representation inherent in GSplats for efficient collision-checking and corridor generation. We present quantitative and qualitative results on a scene trained

with real images, highlighting the effectiveness of Cine-NeRF. In future work, we seek to deeply explore the more artistic requirements of generative cinematography, like occlusions and lighting.

REFERENCES

[1] Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, and Mac Schwager. Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):4606–4613, 2022.

[2] Rogerio Bonatti, Wenshan Wang, Cherie Ho, Aayush Ahuja, Mirko Gschwindt, Efe Camci, Erdal Kayacan, Sanjiban Choudhury, and Sebastian Scherer. Autonomous aerial cinematography in unstructured environments with learned artistic decision-making. *Journal of Field Robotics*, 37(4):606–641, 2020. doi: https://doi.org/10.1002/rob.21931. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21931.

[3] Timothy Chen, Preston Culbertson, and Mac Schwager. Catnips: Collision avoidance through neural implicit probabilistic scenes. *arXiv preprint arXiv:2302.12931*, 2023.

[4] Timothy Chen, Ola Shorinwa, Weijia Zeng, Joseph Bruno, Philip Dames, and Mac Schwager. Splat-Nav: Safe real-time robot navigation in Gaussian splatting maps. *arXiv preprint*, 2024.

[5] Marc Christie, Rumesh Machap, Jean-Marie Normand, Patrick Olivier, and Jonathan Pickering. Virtual camera planning: A survey. In Andreas Butz, Brian Fisher, Antonio Krüger, and Patrick Olivier, editors, *Smart Graphics*, pages 40–52, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31905-4.

[6] Herbert Edelsbrunner. Surface Reconstruction by Wrapping Finite Sets in Space. In Boris Aronov, Saugata Basu, János Pach, and Micha Sharir, editors, *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*, Algorithms and Combinatorics, pages 379–404. Springer, Berlin, Heidelberg, 2003. ISBN 978-3-642-55566-4.

[7] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989. ISSN 1558-0814.

[8] Igor Gilitschenski and Uwe D. Hanebeck. A robust computational test for overlap of two arbitrary-dimensional ellipsoids in fault-detection of kalman filters. In *2012 15th International Conference on Information Fusion*, pages 396–401, 2012.

[9] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *2011 IEEE international conference on robotics and automation*, pages 1478–1483. IEEE, 2011.

[10] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[11] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[12] Justin* Kerr, Chung Min* Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *International Conference on Computer Vision (ICCV)*, 2023.

[13] Pileun Kim, Jingdao Chen, and Yong K Cho. Slam-driven robotic mapping and registration of 3d point clouds. *Automation in Construction*, 89:38–48, 2018.

[14] SM LaValle. Planning algorithms. *Cambridge University Press google schola*, 2:3671–3678, 2006.

[15] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[16] Sikang Liu, Michael Watterson, Kartik Mohta, Ke Sun, Subhrajit Bhattacharya, Camillo J. Taylor, and Vijay Kumar. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3D complex environments. *IEEE Robotics and Automation Letters (RA-L)*, 2(3):1688–1695, 2017. doi: 10.1109/LRA.2017.2663526.

[17] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. *arXiv preprint arXiv:2312.06741*, 2023.

[18] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020.

[19] Thomas Müller. tiny-cuda-nn, 4 2021. URL https://github.com/NVlabs/tiny-cuda-nn.

[20] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41 (4):102:1–102:15, July 2022.

[21] Stanley Osher and Ronald P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, New York, 2003. ISBN 978-0-387-22746-7.

[22] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. *arXiv preprint arXiv:2312.16084*, 2023.

[23] Jack Sherman. Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *Annals of mathematical statistics*, 20(4):621, 1949.

[24] M. Tancik, E. Weber, R. Li, B. Yi, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, A. Kanazawa, and E. Ng. Nerfstudio: A framework for neural radiance field development. In *SIGGRAPH*, 2023.

[25] Chi Yan, Delin Qu, Dong Wang, Dan Xu, Zhigang Wang, Bin Zhao, and Xuelong Li. GS-SLAM: Dense

visual slam with 3d gaussian splatting. *arXiv preprint arXiv:2311.11700*, 2023.

[26] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R Oswald. Gaussian-SLAM: Photo-realistic dense slam with gaussian splatting. *arXiv preprint arXiv:2312.10070*, 2023.