Real-world Visual Navigation in a Simulator: A New Benchmark

Henghui Bao*, **Kiran Lekkala***, Laurent Itti
University of Southern California
Los Angeles, CA 90089, USA

Abstract

In this paper, we explore advanced techniques in novel view rendering, particularly Gaussian Splatting, to create a simulator using a large-scale outdoor dataset. Our simulator, Beogym, is data-driven and built from data collected using a mobile robot. Our proposed pipeline processes the dataset to obtain an interconnected sequence of Gaussian splat files. These are then used by an engine to load appropriate splat files and render image frames during simulation. Beogym offers first-person view imagery, facilitating realistic training environments that could be used for enhancing and evaluating the learning capabilities of autonomous agents for visual navigation. It incorporates a sophisticated motion model and a sequence graph for seamless querying and loading of different sectors of the environment. The result closely resembles real-world navigation through smooth transitions across splat files.

1. Introduction

Learning to navigate without map is a task designed to enable agents to mimic human-like goal-oriented behaviors, relying solely on visual observations. Simulators are widely used in practice to seamlessly enable the agent to learn such behaviors. However, many recent works [1] attempt to fulfill only a few aspects out of the following; simulators that have photorealistic rendering, high performance, efficient utilization of compute resources and real-world transferability. Our method aims fulfill all the above requirements through advanced techniques in novel view rendering, such as Neural Radiance Fields (NeRF) [5] or Gaussian Splatting [3]. By interpolating these intermediate views, we seek to develop a real-time simulator that would not only facilitate more effective learning and navigation for robotic agents but also enhance their applicability in real-world environments.

*Equal Contribution, Corresponding author: henghuib@usc.edu
This work was supported by the National Science Foundation (award 2318101), C-BRIC (one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA) and the Army Research Office (W911NF2020053). The authors affirm that the views expressed herein are solely their own, and do not represent the views of the United States government or any agency thereof.

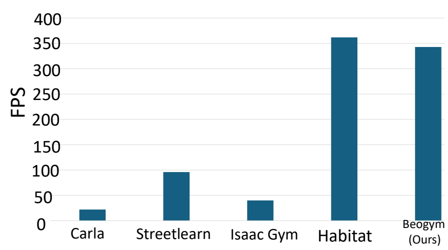

Figure 1. **Comparision with other simulators.** Rendering speed or Frames per Second (FPS) recorded for a single thread process with frame resolution of 1280×720, single episode ~ 200 timesteps.

2. Proposed Simulator

We propose *Beogym*, a real-time simulator that allows an autonomous agent capable of navigating in an environment. After the agent executes an action/control signal $\mathbf{u_t}$, given an input image $\mathbf{I_t}$ the simulator computes the pose $\mathbf{x_{t+1}}$ at the next timestep using a motion model and renders an image, by querying from the splat file. One of the crucial aspects of a simulator apart from realistic quality is high performance in terms of rendering speed. Our method ensures that the visual feedback provided to the agent is both realistic and computationally efficient, facilitating more effective training and navigation. We compare the performance of our simulator with other SoTA simulators as shown in Figure 2.

2.1. Gaussian splat based rendering

The dataset that we use for our simulator consists of images and pointcloud data collected on a mobile robot that is collected across the USC campus [4]. In our methodology, each *sector*, defined as a splat file that is trained using a segmented portion of a trajectory. To optimize the efficiency and performance of reconstruction, we subdivided each session into several sectors, each containing 1250 im-

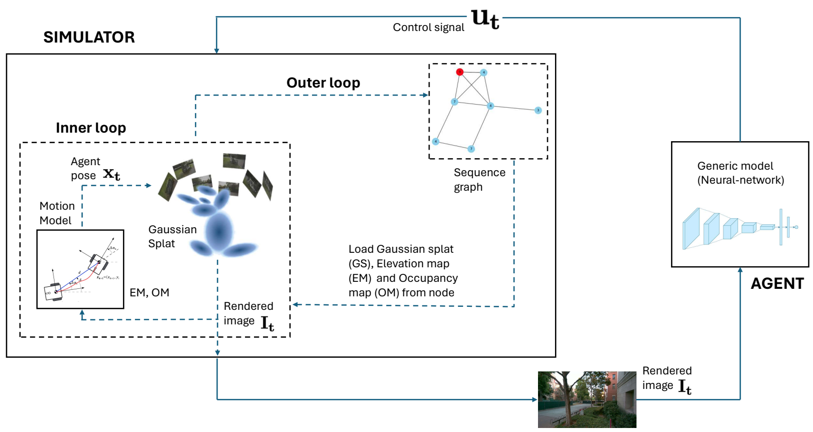

Figure 2. **Overview of our simulator.** The agent obtains a percept/image from the simulator and estimates a control signal $\mathbf{u_t}$. The outer loop in the simulator determines whether the agent has passed the boundaries of the current *sector* and if a new splat file has to be loaded using the sequence graph. The inner loop corresponds to the motion model that computes the pose $\mathbf{x_t}$ that is then used to render the percept in the next timestep using the gaussian splat file.

ages captured from the five cameras. For every sector, we utilize the collection of images and use COLMAP [7] to automatically obtain poses. This pose-annoted set of images is passed as an input for training a splat file using Gaussian Splatting [3]. The result would then be obtained as an explicit 3D representation of a specific scene or a *sector*. Subsequently, we store these splat files, which are later used for rendering purposes.

2.2. Sequence Graph for querying splat files

To facilitate navigation between sectors, we constructed a sequence graph based on discrete image poses. In this graph, each sector serves as a node corresponding to a splat file and other related metadata. The connectivity between sectors is represented by an edge that corresponds to a transformation matrix $\mathbf{T_a^b}$ from a splat file a to the other b.

As the agent navigates within the simulation environment, a key element of realism is its interaction between different sectors. When the agent approaches the boundary of its current sector, the simulator recognizes this transition and initiates the process of loading the appropriate splat file for the new sector. This mechanism ensures a seamless visual experience as the agent moves through diverse parts of the simulated environment.

2.3. Motion model

In the *Beogym* simulator, given a specific control signal, the agent's subsequent pose $\mathbf{x_t}$ at timestep t is governed by the motion model. This model is crucial for simulating realistic navigation behaviours, akin to those exhibited by actual robots in real-world scenarios. One key challenge addressed in our model is the maintenance of consistent height, raw and yaw orientation relative to the ground by the agent. This is crucial for ensuring realism and practical applicability, especially environments where the terrain may vary, potentially leading to the agent ”floating” above the ground or colliding with terrain features. To address this, we employ a sophisticated method to compute the z-axis component or the elevation of the agent's pose accurately, using elevation and occupancy maps obtained from the LiDAR data.

Furthermore, our simulation is designed to be adaptable and responsive. An agent can be initialized at any location within the simulation environment determined by the occupancy map. As the agent moves, guided by the motion model, its pose is continually updated, and the rendering process adapts accordingly, thus creating a seamless and continuous simulation experience. This iterative process, where the motion model predicts the next pose and the simulator renders the new view, forms the core inner-loop of our simulation as shown in Figure 2.

As stated before, Within each sector, we employ COLMAP [6] to obtain image poses that are then used for training a splat file. However, it's important to note that these poses do not represent ground-truth poses obtained from the LiDAR sensor, and merely are used for training a Gaussian splat file. To utilize elevation maps derived from ground-truth point clouds, we must transform the coordinate system of the poses from COLMAP to those of the elevation maps. We employ the *Kabsch* algorithm [2] to compute this coordinate transformation. The ground truth poses obtained from LiDAR in each sector are less dense than image data and we pair these poses with images having the closest timestamps. After forming all pairs, the *Kabsch* algorithm is executed to derive optimal translation and rotation matrices that minimize the root mean square (RMS) deviation between the two sets of points.

References

[1] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1

[2] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976. 2

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139:1–139:14, 2023. 1, 2

[4] Laurent Itti Kiran Lekkala*, Henghui Bao*. Uscilab3d: A large-scale, long-term, semantically annotated outdoor dataset, 2024. 1

[5] 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 *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, pages 405–421. Springer, 2020. 1

[6] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4104–4113. IEEE Computer Society, 2016. 2

[7] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2