NeRF-SLAM Docker

Sarvesh Mayilvahanan, Hersh Vakharia, Jack Fenton, Walter Xu
{smayil, hershv, fentonj, yunongxu}@umich.edu

***Abstract*—Simultaneous localization and mapping (SLAM) is a fundamental robotics problem, which has seen a wide range of solutions. Performing visual SLAM with just monocular images and achieving 3D reconstruction is an extremely challenging problem. NeRF-SLAM achieves this through a combination of dense monocular SLAM, probabilistic volumetric fusion, and real-time hierarchical volumetric neural radiance fields. This scene reconstruction pipeline has shown to be geometrically and photometrically more accurate than other methods. We present a docker environment than allows NeRF-SLAM to work immediately for easy development and deployment. Additionally, we reproduce the results of NeRF-SLAM on the Replica dataset and perform inference on real-world custom datasets.**

Our code is publicly available at https://github.com/sarveshmayil/NeRF-SLAM-docker. Our datasets and video results are available at https://drive.google.com/drive/folders/1QP0KsM6KKRtO_wGg9mvceJ_WIQCdat15

I. Introduction

NeRF-SLAM [1] proposes a scene reconstruction pipeline that combines the benefits of dense monocular SLAM, probabilistic volumetric fusion, and hash-based hierarchical volumetric fusion radiance fields to estimate geometrically (depth) and photometrically (level of detail) accurate maps of 3D scenes in real-time, without the need of depth images or poses.

While Droid-SLAM [2], a state-of-the-art dense monocular SLAM system, utilizes optical flow and probabilistic methods to produce geometrically accurate scene reconstructions, due to the limitations of TSDF representations, their reconstructions lack photometric detail and are not fully complete. Neural radiance fields [3] (NeRFs) on the other hand, offer detailed and accurate scene reconstructions while maintaining multi-view consistency, but are limited by their long convergence times. With the above strengths and limitations in mind, the authors introduce a system consisting of two modules. The first is Droid-SLAM which serves as the tracking module: estimating dense depth maps, camera poses, and providing uncertainty estimates for both depth and poses. Second, for mapping the authors adopt Instant-NGP [4], a hash-based hierarchical volumetric representation that speeds up NeRF training and convergence times by several orders of magnitudes, which is a key enabler of real-time scene reconstruction. The two modules are connected by probabilistic volumetric fusion (σ-Fusion) [5], one of the authors' earlier works, that optimizes depth uncertainties based on the information matrix of the underlying bundle adjustment problem in SLAM. Providing accurate weightings of the depth map from DROID-SLAM allows for the inherent noise in a depth reconstruction to be accounted for in Instant-NGP.

NeRF-SLAM's exceptional accuracy and fast computation time positions it as one of the best scene construction pipelines that can be utilized in a variety of applications. Accordingly, we first present a docker environment that allows easy replication and application of NeRF-SLAM onto different platforms and projects. We then provide some utilities that facilitate the use of custom datasets with NeRF-SLAM. Lastly, we extend the NeRF-SLAM's evaluation using a real world custom dataset to provide further insight into its performance.

II. Related Works

The main problems NeRF-SLAM addresses are both photometric and geometric accuracy of scene reconstructions and NeRF training speed for real-time applications, particularly in the case of a monocular RGB camera with no additional sensor modalities. The dense monocular SLAM module in NeRF-SLAM addresses the former while the NeRF module addresses the latter. We outline several related works that have made contributions in these domains.

iMAP [6] shows that MLPs can be used as the only scene representation in a real-time SLAM system for an RGB-D camera. It achieves impressive accuracies and real-time mapping and tracking. iMAP proves that using an implicit MLP over standard dense SLAM methods can result in faster estimations of unobserved and ambiguous regions. Their method however, is enabled by depth measurements from the camera. As depth measurements in RGB-D cameras are more prone to errors and are generally more expensive, NeRF-SLAM seeks to remove the dependency on depth measurements.

iNeRF [7] presents a mesh-free pose estimation framework that produces highly accurate pose estimations from RGB images by "inverting" a NeRF. It is highly generalizable as the NeRF can learn a continuous representation of different scenes and viewpoints. The utilization of a NeRF also brings high scalability of localization in large scenes. However, the main drawback of iNeRF is that the NeRF must be trained, introducing the need for additional data and computation, making iNeRF unsuitable for real-time applications.

Orbeez-SLAM [8] addresses the computational performance issues of learning based SLAM methods such as iNeRF by utilizing visual odometry to create dense maps from RGB images. Camera poses are obtained through optimizations that minimize reprojection errors. Orbeez-SLAM achieves superior accuracies and computation times compared to many state of the art methods, however, its accuracy is limited by the drawbacks of the nonlinear optimizations. As the optimizations rely

heavily on initial pose estimates, the loss of depth information from using monocular cameras cause significant performance degradation.

Finally, NeRF-SLAM builds upon the advantages and addresses the drawbacks of some of the aforementioned methods. It leverages a tracking module of Droid-SLAM to provide poses, depths, and uncertainties, a fusion module of probabalistic volumetric fusion to optimizes depths, and a mapping module that uses Instant-NGP to further speed up NeRF training. The end result is a end-to-end pipeline that provides accurate scene reconstructions from monocular images in real time.

III. Docker Setup

One of the main goals of this project was to provide a development and testing environment capable of running NeRF-SLAM. This was accomplished by creating a docker image with all dependencies and requirements pre-installed and ready to use.

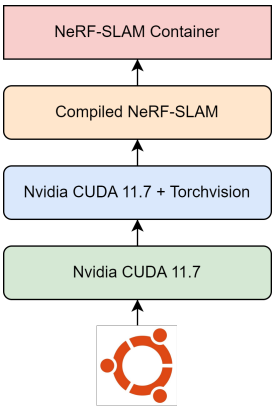

Fig. 1. NeRF-SLAM docker architecture.

As seen in Figure 1, the docker structure used for the NeRF-SLAM image has three main layers. The base image is a publicly available official Nvidia CUDA 11.7 image built on Ubuntu 20.04. From this base image, we install some commonly useful packages (such as python) as well as the correct version of torchvision corresponding to CUDA 11.7. The NeRF-SLAM image is built on this CUDA image and installs all the required packages needed to compile and run NeRF-SLAM (not all of which was included in the original repository's requirements). The local version of NeRF-SLAM is copied into the image and compiled completely after which the image can be run as a container and used to run NeRF-SLAM.

There were a number of additions and changes that had to be made in order for NeRF-SLAM to compile and run correctly. This included obtaining the correct version of packages such as *cmake* and move to the correct branches of packages such as *gtsam* and *instantngp*. Overall, we found that building all images took approximately 60-70 min.

When running the image as a docker container, X-forwarding and access to local GPUs are enabled and all processes are run interactively so that the GUI can be seen and interacted with locally.

IV. Reproduction

With an environment set up to run NeRF-SLAM, we set out to reproduce the results from the original NeRF-SLAM paper. We chose to perform NeRF-SLAM on one of the key data sequences in the paper, which was `room0` from the Replica dataset [9]. In Figure 2, we show the qualitative results comparing the constructed NeRF to the original image for `room0`.

(a) Real

(b) NeRF-SLAM

(c) Ours

Fig. 2. Qualitative results on `room0` of Replica dataset comparing the real and recreated images from [1] and ours.

Comparing the real and reconstructed images for `room0`, we see that NeRF-SLAM is able to recreate the room very accurately in both color and depth.

We used an Nvidia RTX 4080 GPU with 12 GB of VRAM to produce these results.

(a) Real (b) NeRF-SLAM

Fig. 3. Qualitative results comparing real image to recreated NeRF on recorded video of UM FRB kitchenette.

V. Extension

Our extension of NeRF-SLAM involves creating the docker setup, as explained in Section III, that will allow others to more easily experiment with the NeRF-SLAM codebase. Additionally, we ran NeRF-SLAM on our own datasets collected in the University of Michigan Ford Robotics Building (FRB). Fig. 3 shows our results from data collected in a kitchenette, and Fig. 4 shows our results from data collected around the University of Michigan Autonomous Robotic Vehicle Team's robot. These videos were recorded in 1920×1080 resolution, but the frames were resized to 640×360.

NeRF-SLAM requires datasets to have the individual frames of the video, as well as a `transforms.json` file that contains the camera intrinsics, image file paths, optional depth-image file paths, and camera transform information. It was unclear why the system needed camera transforms for each frame, but we set all transforms to the identity matrix, and the system was still able to track the camera position. The original repository does not provide any method of running off of a video, so we wrote a script that generates the frames and JSON file from a video. Since our videos were collected on a simple monocular cell phone camera, we did not provide any depth images.

Our results show that NeRF-SLAM is able to successfully track monocular camera poses and generate a NeRF of the surroundings. The kitchenette example worked particularly well as shown in in Figure 3. This is likely due to it being a fully enclosed space, which makes it easy to infer depth from the images. While the robot example in Figure 4 was able to make a decent rendering of the robot itself, since it was in an open space, the surroundings were less defined.

The need for frames with varying heights and angles was also apparent in our tests. The resulting NeRFs would look quite poor from certain angles, as the video may not have captured the scene well from that angle. Therefore, to make a NeRF of an object, like the robot, it was important that we took the video all around the robot at varying heights.

(a) Real (b) NeRF-SLAM

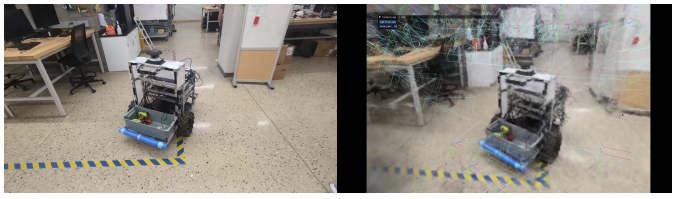

Fig. 4. Qualitative results on recorded video of UM-ARV robot.

VI. Discussion & Conclusion

In the original NeRF-SLAM paper, the authors claim that they achieved real-time performance of 12 frames per second at 640×480 resolution with 11 GB of VRAM. However, we were consistently running into VRAM constraints, even with the 12 GB of VRAM our system has. We would have to increase the image stride and lower the buffer from their suggested values

in order to run a sequence without the system crashing. The stride affects the number of images loaded into VRAM and the buffer defines the number of keyframes stored and used. We found that increasing the stride such that the system loads approximately 250 images worked well. It was difficult to tell if the system was running real-time at 12 frames per second, though it qualitatively felt slower.

To conclude, NeRF-SLAM is able to successfully track poses and build a NeRF of the surroundings given just monocular images. Our results show fairly high fidelity renderings; it especially succeeds in full enclosed spaces. However, future work must be done to address the limitations of the system in terms of memory consumption.

REFERENCES

[1] A. Rosinol, J. J. Leonard, and L. Carlone, "Nerf-slam: Real-time dense monocular slam with neural radiance fields," *arXiv preprint arXiv:2210.13641*, 2022.

[2] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in neural information processing systems*, vol. 34, pp. 16 558–16 569, 2021.

[3] 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.

[4] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022.

[5] A. Rosinol, J. J. Leonard, and L. Carlone, "Probabilistic volumetric fusion for dense monocular slam," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 3097–3105.

[6] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "imap: Implicit mapping and positioning in real-time," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6229–6238.

[7] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "inerf: Inverting neural radiance fields for pose estimation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1323–1330.

[8] C.-M. Chung, Y.-C. Tseng, Y.-C. Hsu, X.-Q. Shi, Y.-H. Hua, J.-F. Yeh, W.-C. Chen, Y.-T. Chen, and W. H. Hsu, "Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping," *arXiv preprint arXiv:2209.13274*, 2022.

[9] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.