

Loc-NeRF: Monte Carlo Localization using Neural Radiance Fields

Dominic Maggio, Marcus Abate, Jingnan Shi, Courtney Mario, Luca Carlone

Abstract—We present Loc-NeRF, a real-time vision-based robot localization approach that combines Monte Carlo localization and Neural Radiance Fields (NeRF). Our system uses a pre-trained NeRF model as the map of an environment and can localize itself in real-time using an RGB camera as the only exteroceptive sensor onboard the robot. While neural radiance fields have seen significant applications for visual rendering in computer vision and graphics, they have found limited use in robotics. Existing approaches for NeRF-based localization require both a good initial pose guess and significant computation, making them impractical for real-time robotics applications. By using Monte Carlo localization as a workhorse to estimate poses using a NeRF map model, Loc-NeRF is able to perform localization faster than the state of the art and without relying on an initial pose estimate. In addition to testing on synthetic data, we also run our system using real data collected by a Clearpath Jackal UGV and demonstrate for the first time the ability to perform real-time and global localization (albeit over a small workspace) with neural radiance fields. We make our code publicly available at <https://github.com/MIT-SPARK/Loc-NeRF>.

I. INTRODUCTION

Vision-based localization is a foundational problem in robotics and computer vision, with applications ranging from self-driving vehicles [1] to robot manipulation [2]. Classical approaches for camera pose estimation typically address the task by adopting a multi-stage paradigm, where keypoints are first detected and matched between each frame and the map (where the latter is stored as a collection of images with the corresponding keypoints and descriptors), and six degree-of-freedom (DoF) poses are estimated using Perspective-n-Point (PnP) algorithms [2], [3], [4]. However, such methods are sensitive to the quality of the keypoint matching and require storing a database of images as the map representation.

Orthogonal to the camera pose estimation literature, advances in deep learning have led to a plethora of works investigating implicit shape and scene representations [5], [6], [7], [8], [9]. In particular, Neural Radiance Fields (NeRF) have gained significant popularity, as they can encode both 3D geometry and appearance of an environment [10]. NeRFs are fully-connected neural networks trained using a collection

D. Maggio is with the Laboratory for Information & Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA, and is a Draper Scholar with the Perception and Embedded ML Group, Draper, Cambridge, MA, USA, drmaggio@mit.edu

M. Abate, J. Shi, and L. Carlone are with the Laboratory for Information & Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA, {[mabate](mailto:mabate@mit.edu), [jnshi](mailto:jnshi@mit.edu), [lcarlone](mailto:lcarlone@mit.edu)}@mit.edu

C. Mario is with the Draper Perception and Embedded ML Group, Draper, Cambridge, MA, USA, cmario@draper.com

This work was partially funded by the NASA Flight Opportunities under grant Nos 80NSSC21K0348, ARL DCIST CRA W911NF-17-2-0181, and an Amazon Research Award.



Fig. 1. Real-time experiments with Loc-NeRF using a Clearpath Jackal UGV (left) equipped with a Realsense d455 camera. Examples of NeRF renderings near the beginning, middle, and end of the experiment (right).

of monocular images to approximate functions taking 3D positions as inputs and returning RGB values and view density (the so called “radiance”) as output. NeRF can then be used in conjunction with ray tracing algorithms to synthesize novel views [10]. NeRF has even been extended to address challenging rendering problems involving non-Lambertian surfaces, variable lighting conditions [11], and motion blur [12].

If we view NeRF as a function that encodes spatial and radiance information, a natural question that arises is: *can we leverage advances in NeRF to solve localization tasks for robotics?* The existing literature on NeRF-based localization is sparse. Yen-Chen *et al.* [8] propose iNeRF, the first method to demonstrate pose estimation by “inverting” a NeRF; iNeRF estimates the camera pose by performing local optimization of a loss function quantifying the per-pixel mismatch between the map and a given camera image. Adamkiewicz *et al.* [13] propose NeRF-Navigation, which demonstrates the possibility of using NeRF as a map representation across the autonomy stack, from state estimation to planning.

Contributions. Following the same research thrust as iNeRF and NeRF-Navigation, we present *Loc-NeRF*, a 6DoF pose estimation pipeline that uses a (particle-filter-based) Monte Carlo localization [14] approach as a novel way to extract poses from a NeRF. More in detail, we design a vision-based particle-filter localization pipeline, that (i) uses NeRF as a map model in the update step of the filter, and (ii) uses visual-inertial odometry or the robot dynamics for highly accurate motion estimation in the prediction step of the filter. The proposed particle-filter approach allows pose

estimation with poor or no initial guess, while allowing us to adjust the computational effort by modifying the number of particles. We present experiments showing that Loc-NeRF can: (i) estimate the pose of a single image without relying on an accurate initial guess, (ii) perform global localization, and (iii) achieve real-time tracking with real-world data (Fig. 1).

The rest of the paper is organized as follows. Section II discusses related work. Section III provides a high level overview of NeRF. Section IV presents the structure of Loc-NeRF. Section V evaluates Loc-NeRF on three types of experiments: benchmarking with iNeRF on pose estimation from a single image, benchmarking with NeRF-Navigation on simulated drone flight data, and real-time navigation with real-world data. Finally, Section VI concludes the paper.

II. RELATED WORK

Neural Implicit Shape Representations. Shape representations are central to many problems in computer vision, computer graphics [15], [16], and robotics [17], [18]. Traditional shape representations such as points clouds, meshes, and voxel-based models, while being well studied and commonly used in robotics, still suffer from several drawbacks. For example, point clouds lack the ability to encode surface information. Meshes encode surfaces, but it remains challenging to estimate highly accurate meshes from sensor data collected by a robot [19]. Similarly, the accuracy of voxel-based models is intrinsically limited by the voxel size used for discretization.

Recently, neural implicit shape representations have been developed as effective alternatives to traditional shape representations [7], [20], [21], [16]. Park *et al.* [7] represent shapes as a learned signed distance function using fully-connected neural networks. Mescheder *et al.* [20] learn a probability representation of occupancy grids to represent surfaces. Mildenhall *et al.* [10] propose NeRF and show that by adding view directions as additional inputs, it is possible to train a network to synthesize novel and photo-realistic views.

Additional studies have investigated the problem of training NeRF with images whose poses are either unknown or known with low accuracy [22], [23], [24], [25]. These methods take several hours or over a day to train and are intended for building a NeRF as opposed to real-time pose estimation with a trained NeRF. NeRF has also been extended to large-scale [26], [27], [28] and unbounded scenes [29], [30], which has the potential to enable neural representations of large-scale scenes such as the ones typically encountered in robotics applications, from drone navigation to self-driving cars.

Slow training and rendering time has been a longstanding challenge for NeRF, with several recent works proposing computational enhancements. Müller *et al.* [31] use a multi-resolution hash encoding to train a NeRF in seconds and render images on the order of milliseconds. Additionally, some works have utilized depth information to improve rendering time [32], [33], and training time [34], [35], [36]. Related to using depth, Clark [37] uses a volumetric dynamic B+Tree data structure to achieve real-time scene reconstruction and Yu *et al.* [38] use a scene representation based on octrees.

Visual Localization. Classical approaches for visual localization and SLAM in robotics typically use a multi-stage paradigm, where some sparse representations (such as keypoints) are used to enable tracking and localization [39], [40], [41]. In some works, instead of sparse keypoints, a dense representation is used to represent the 3D environment [42]. The backend of classical localization methods typically rely on well established estimation-theoretic techniques, such as maximum a posteriori (optimization-based) estimation, Kalman filters, particle filters, and grid-based histogram filters; these techniques enable tracking the pose of the robot over time [43].

More recently, localization and mapping have been studied in conjunction with neural implicit representations. Sucar *et al.* [9] propose iMAP, which demonstrates that an MLP can be used to represent the scene in simultaneous localization and mapping. Zhu *et al.* [6] develop NICE-SLAM, which extends the idea of MLP-based scene representation to larger, multi-room environments. Ortiz *et al.* [44] propose iSDF, which is a continual learning system for real-time signed distance field reconstruction. iMAP, NICE-SLAM, and iSDF utilize depth information from a stereo camera in addition to color images. In the RGB-only case, the literature on robot localization based on neural implicit representations is still sparse. Yen *et al.* [8] develop iNeRF, which estimates the pose of a provided image given a trained NeRF model and an initial pose guess by optimizing a photo-metric loss using back-propagation with respect to the pose; iNeRF requires a good initial guess and the optimization entails a high computational overhead. Adamkiewicz *et al.* [13] propose NeRF-Navigation, which uses NeRF to power the entire autonomy stack of a drone, including estimation, control, and planning. Similar to iNeRF, NeRF-Navigation optimizes a loss that includes a photometric loss along a process loss term induced by the robot dynamics and control actions. This added process loss enables tracking a path across multiple images, but the method still requires a good initial guess and incurs a high computation overhead similar to iNeRF. In this paper, we improve upon iNeRF and NeRF-Navigation and propose Loc-NeRF. The particle filter backbone [14] of Loc-NeRF allows relaxing the reliance on a good initial estimate to bootstrap localization.

III. NeRF PRELIMINARIES

NeRF [10] uses a multilayer perceptron (MLP) to store a radiance field representation of a scene and render novel viewpoints. NeRF is trained on a scene given a set of RGB images with known poses and a known camera model. At inference time, NeRF renders novel views by predicting the density σ and RGB color c of a point in 3D space given the 3D position and viewing direction of the point. To predict the RGB value of a single pixel, NeRF projects a ray r from the center point of the camera, through a pixel in the image plane. Then n_{coarse} samples are uniformly generated along the ray and n_{fine} samples are selected based on the estimated σ of the coarse samples. Volume rendering is then used to

estimate the color value $\mathcal{C}(\mathbf{r})$ for the pixel:

$$\mathcal{C}(\mathbf{r}) = \int_{z_{\text{near}}}^{z_{\text{far}}} T(\mathbf{r}, z) \sigma(\mathbf{r}, z) \mathbf{c}(\mathbf{r}, z) dz \quad (1)$$

where z_{near} and z_{far} are bounds on the sampled depth z along the ray \mathbf{r} and $T(\mathbf{r}, z)$ is given by:

$$T(\mathbf{r}, z) = \exp\left(-\int_{z_{\text{near}}}^z \sigma(\mathbf{r}, z') dz'\right) \quad (2)$$

The reader is referred to [10] for a more detailed description.

IV. LOC-NeRF: MONTE CARLO LOCALIZATION USING NEURAL RADIANCE FIELDS

We now present Loc-NeRF, a real-time Monte Carlo localization method that uses NeRF as a map representation. Given a map \mathcal{M} (encoded by a trained NeRF), RGB input image \mathcal{I}_t at each time t , and motion estimates \mathcal{O}_t between time $t-1$ and time t , Loc-NeRF estimates the 6DoF pose of the robot \mathbf{X}_t at time t . In particular, Loc-NeRF uses a particle filter to estimate the posterior probability $\mathbb{P}(\mathbf{X}_t | \mathcal{M}, \mathcal{I}_{1:t}, \mathcal{O}_{1:t})$, where $\mathcal{I}_{1:t}$ and $\mathcal{O}_{1:t}$ are the sets of images and motion measurements collected between the initial time 1 and the current time t , respectively.

Monte Carlo localization [14] relies on a particle filter and models the posterior distribution $\mathbb{P}(\mathbf{X}_t | \mathcal{M}, \mathcal{I}_{1:t}, \mathcal{O}_{1:t})$ as a weighted set of n particles:

$$S_t = \{\langle \mathbf{X}_t^i, w_t^i \rangle \mid i = 1, \dots, n\} \quad (3)$$

where \mathbf{X}_t^i is a 3D pose (represented as a 4×4 transformation matrix in our implementation) associated to the i -th particle, and $w_t^i \in [0, 1]$ is the corresponding weight. The particle filter then updates the set of particles at each time instant (as new images and odometry measurements are received) by applying three steps: prediction, update, and resampling.

A. Prediction Step

The prediction step predicts the set of particles S_t at time t from the corresponding set of particles S_{t-1} at time $t-1$, given a measurement \mathcal{O}_t of the robot motion between time $t-1$ and time t ; the measurement is typically provided by some odometry source (*e.g.*, wheel or visual odometry) or obtained by integrating the robot dynamics; in our implementation, we either use visual-inertial odometry or integrate the robot dynamics, depending on the experiment. When a measurement of the robot's relative motion \mathcal{O}_t is received, the set of particles can be updated by sampling new particles using the motion model $\mathbb{P}(\mathbf{X}_t | \mathbf{X}_{t-1}, \mathcal{O}_t)$. While the particle filter can accommodate arbitrary motion models, here we adopt a simple model that updates the pose of each particle according to the motion \mathcal{O}_t and then adds Gaussian noise to account for odometry errors:

$$\mathbf{X}_t = \mathbf{X}_{t-1} \cdot \mathcal{O}_t \cdot \mathbf{X}_\epsilon \quad , \quad \mathbf{X}_\epsilon = \text{Exp}(\boldsymbol{\delta}), \quad (4)$$

where \mathbf{X}_ϵ is the prediction noise, $\text{Exp}(\cdot)$ is the exponential map for SE(3) (the Special Euclidean group), and $\boldsymbol{\delta} \in \mathbb{R}^6$ is a normally distributed vector with zero mean and covariance $\text{diag}(\sigma_R^2 \cdot \mathbf{I}_3, \sigma_t^2 \cdot \mathbf{I}_3)$, where σ_R and σ_t are the rotation and translation noise standard deviations, respectively.

B. Update Step

The update step uses the camera image \mathcal{I}_t collected at time t to update the particle weights w_t^i . According to standard Monte Carlo localization [14], we update the weights using the measurement likelihood $\mathbb{P}(\mathcal{I}_t | \mathbf{X}_t^i, \mathcal{M})$, which models the likelihood of taking an image \mathcal{I}_t from pose \mathbf{X}_t^i in the map \mathcal{M} . We use a heuristic function to approximate the measurement likelihood as follows:

$$w_t^i = \left(\frac{M}{\sum_{j=1}^M (\mathcal{I}_t(\mathbf{p}_j) - \mathcal{C}(\mathbf{r}(\mathbf{p}_j, \mathbf{X}_t^i)))^2} \right)^4 \quad (5)$$

where $\mathbf{r}(\mathbf{p}_j, \mathbf{X}_t^i)$ computes the ray emanating from pixel \mathbf{p}_j when the robot is at pose \mathbf{X}_t^i , and $\mathcal{I}_t(\mathbf{p}_j)$ is the image intensity at pixel \mathbf{p}_j . Intuitively, eq. (5) compares the collected image \mathcal{I}_t with the image $\mathcal{C}(\mathbf{r})$ predicted by the NeRF map and assigns low weights to particles where the two images do not match. For efficient computation, we compute the weight update (5) only using a subset of M pixels randomly sampled from \mathcal{I}_t . Weights are then normalized to sum up to 1.

C. Resampling Step

After the update step, we resample n particles from the set S_t with replacement, where each particle is sampled with probability w_t^i . As prescribed by standard particle filtering, the resampling step allows retaining particles that are more likely to correspond to good pose estimates while discarding less likely hypotheses.

D. Computational Enhancements and Pose Estimate

Particle Annealing. To improve convergence of the filter and reduce the computational load, we automatically adjust the prediction noise (σ_R, σ_t) and the number of particles n over time. Our particle annealing approach is similar in spirit to the KLD-based approach from [45], but we also adjust the noise statistics. As shown in Section V, this leads to computational and accuracy improvements. The prediction noise and number of particles are updated as shown in Algorithm 1. In particular, we use the standard deviation of the particles' position σ_{S_t} to characterize the spread of the particles in the filter at time t and reduce the prediction noise and the number of particles (initially set to $\sigma_{R,\text{init}}, \sigma_{t,\text{init}}$, and n_{init}) when the spread falls below given thresholds (α_{refine} and $\alpha_{\text{super-refine}}$ in Algorithm 1).

Algorithm 1 Particle Annealing

Input: $\sigma_{R,\text{init}}, \sigma_{t,\text{init}}, \sigma_{S_t}, n_{\text{init}}$

$\sigma_R \leftarrow \sigma_{R,\text{init}}$

$\sigma_t \leftarrow \sigma_{t,\text{init}}$

$n \leftarrow n_{\text{init}}$

if $\sigma_{S_t} < \alpha_{\text{super-refine}}$ **then**

$\sigma_R \leftarrow \frac{\sigma_{R,\text{init}}}{4}, \sigma_t \leftarrow \frac{\sigma_{t,\text{init}}}{4}, n \leftarrow n_{\text{reduced}}$

else if $\sigma_{S_t} < \alpha_{\text{refine}}$ **then**

$\sigma_R \leftarrow \frac{\sigma_{R,\text{init}}}{2}, \sigma_t \leftarrow \frac{\sigma_{t,\text{init}}}{2}, n \leftarrow n_{\text{reduced}}$

else

$\sigma_R \leftarrow \sigma_{R,\text{init}}, \sigma_t \leftarrow \sigma_{t,\text{init}}$

end if

Obtaining a Pose Estimate from the Particles. Besides computing the set of particles, Loc-NeRF returns a single pose estimate $\hat{\mathbf{X}}_t$ that is computed as a weighted average of the particle poses. In particular, the position portion of $\hat{\mathbf{X}}_t$ is simply the weighted average of the positions of the particles in S_t . The rotation portion of $\hat{\mathbf{X}}_t$ is found by solving the geodesic L_2 single rotation averaging problem. The reader is referred to [46] and [47] for details on rotation averaging.

V. EXPERIMENTS

We evaluate Loc-NeRF on three sets of experiments: (i) pose estimation from a single image using the LLFF dataset [48] given either a poor initial guess or no initial guess, where we benchmark against iNeRF [8] (Section V-A), (ii) pose estimation over time using synthetic data from Blender [49], where we benchmark against NeRF-Navigation [13] (Section V-B), and (iii) a full system demonstration where we perform real-time pose tracking using data collected by a Clearpath Jackal UGV (Section V-C).

A. Single-image Pose Estimation: Comparison with iNeRF

Setup. To show Loc-NeRF’s ability to quickly localize given a camera image and from a poor initial guess, we use the same evaluation protocol used in iNeRF [8]. Using 4 scenes (Fern, Fortress, Horns, and Room) from the LLFF dataset [48], we pick 5 random images from each dataset and estimate the pose of each image. For this experiment, both Loc-NeRF and iNeRF use the same pre-trained weights from NeRF-Pytorch [50]. As in [8], we give iNeRF an initial pose guess $\mathbf{X}_{\text{iNeRF}}$. The rotation component of $\mathbf{X}_{\text{iNeRF}}$ is obtained by randomly sampling an axis from the unit sphere and rotating about that axis by a uniformly sampled angle between $[-40^\circ, 40^\circ]$ with respect to the ground truth rotation. The position portion of $\mathbf{X}_{\text{iNeRF}}$ is obtained by uniformly perturbing the ground truth position along each axis by a random amount between $[-0.1 \text{ m}, 0.1 \text{ m}]$. We set iNeRF to use 2048 interest region points ($M = 2048$) as suggested in [8]. Interest regions are found using keypoint detectors and sampling from a dilated mask around those keypoint.

Since Loc-NeRF uses a distribution of particles, we uniformly distribute the initial particles’ poses using:

$$\mathbf{X}_0^i = \mathbf{X}_{\text{iNeRF}} \cdot \text{Exp}(\boldsymbol{\delta}) \quad (6)$$

where the entries corresponding to the rotation component and the translation component of $\boldsymbol{\delta}$ are sampled from a uniform distribution in the range $[-40^\circ, 40^\circ]$ and $[-0.1 \text{ m}, 0.1 \text{ m}]$, respectively. Since we only test on a static image, we set the motion model of Loc-NeRF to be a zero-mean Gaussian distribution whose standard deviation decreases according to Algorithm 1. Loc-NeRF is initialized with 300 particles which reduces to 100 during annealing. We set Loc-NeRF to use 64 ($M = 64$) randomly sampled image pixels per particle.

Results. We plot the fraction of estimated poses with position and rotation error less than 5 cm and 5° in Fig. 2a and Fig. 2b, respectively. Since the computational cost of an iNeRF iteration is different from an iteration of Loc-NeRF (due to number of particles and different values of M) we plot performance against the number of NeRF forward

passes. Loc-NeRF achieves higher accuracy than iNeRF in terms of both position and rotation.

We also plot the average rotation error and average position error for all 20 trials in Fig. 2c and Fig. 2d respectively. In our experiments, the position estimate from iNeRF would occasionally diverge or reach a local minimum and thus the average position error for iNeRF actually increases over time. On a laptop with an RTX 5000 GPU, the update step for Loc-NeRF runs at 0.6 Hz for 300 particles which then accelerates to 1.8 Hz during annealing when the number of particles drops to 100. Loc-NeRF runs approximately 55 seconds per trial. As an ablation study of our annealing process (Algorithm 1), we also include results of Loc-NeRF without annealing. Using annealing shows the most benefit for position accuracy and allows update steps to occur at a faster rate due to the decreased number of particles.

We also demonstrate for the first time that global localization can be performed with NeRF. We repeat a similar experiment as before with LLFF data except now we generate an offset translation by translating the ground truth position along each axis by a random amount between $[-1 \text{ m}, 1 \text{ m}]$ and generate a random distribution of particles in a $2 \times 2 \times 2 \text{ m}$ cube about that offset. We then sample the yaw angle from a uniform distribution in $[-180^\circ, +180^\circ]$, while we initialize the roll and pitch to the ground truth; the latter is done to mimic the setup where we localize using visual-inertial sensors, in which case the IMU makes roll and pitch directly observable. Note that Loc-NeRF still optimizes the particles in a full 6DoF state. We increase the initial number of particles to 600 which drops to 100 during annealing and reduce M to 32. Results of average rotation and translation error from 20 trials are provided in Fig. 3a and Fig. 3b. Loc-NeRF is able to converge to an accurate pose estimate while performing global localization. The annealing process is shown to enable significant improvement for position accuracy and also improves rotation accuracy. iNeRF is unable to produce a valid result for global localization and is thus not included in the figure.

B. Pose Tracking: Comparison with NeRF-Navigation

Setup. NeRF-Navigation [13] performs localization using simulated image streams of Stonehenge recreated in Blender, as if they were collected by a drone flying across the scene (Fig. 4). For this experiment, both Loc-NeRF and NeRF-Navigation use the same pre-trained weights from torch-ngp [51]. We use the same trajectory and sensor images for evaluating Loc-NeRF and NeRF-Navigation. The prediction step for Loc-NeRF uses the same dynamical model estimate of the vehicle’s motion that NeRF-Navigation uses for their process loss. For each image, we run Loc-NeRF for the equivalent number of forward passes as NeRF-Navigation. In particular, we run NeRF-Navigation for 300 iterations per image with $M = 1024$. We use 200 particles for Loc-NeRF with $M = 64$ and run 24 update steps per image.

Results. Fig. 5 shows position and rotation error respectively for a simulated drone course over 18 trials. Note that since NeRF-Navigation uses a similar photometric loss as iNeRF—which requires a good initial guess—we assume

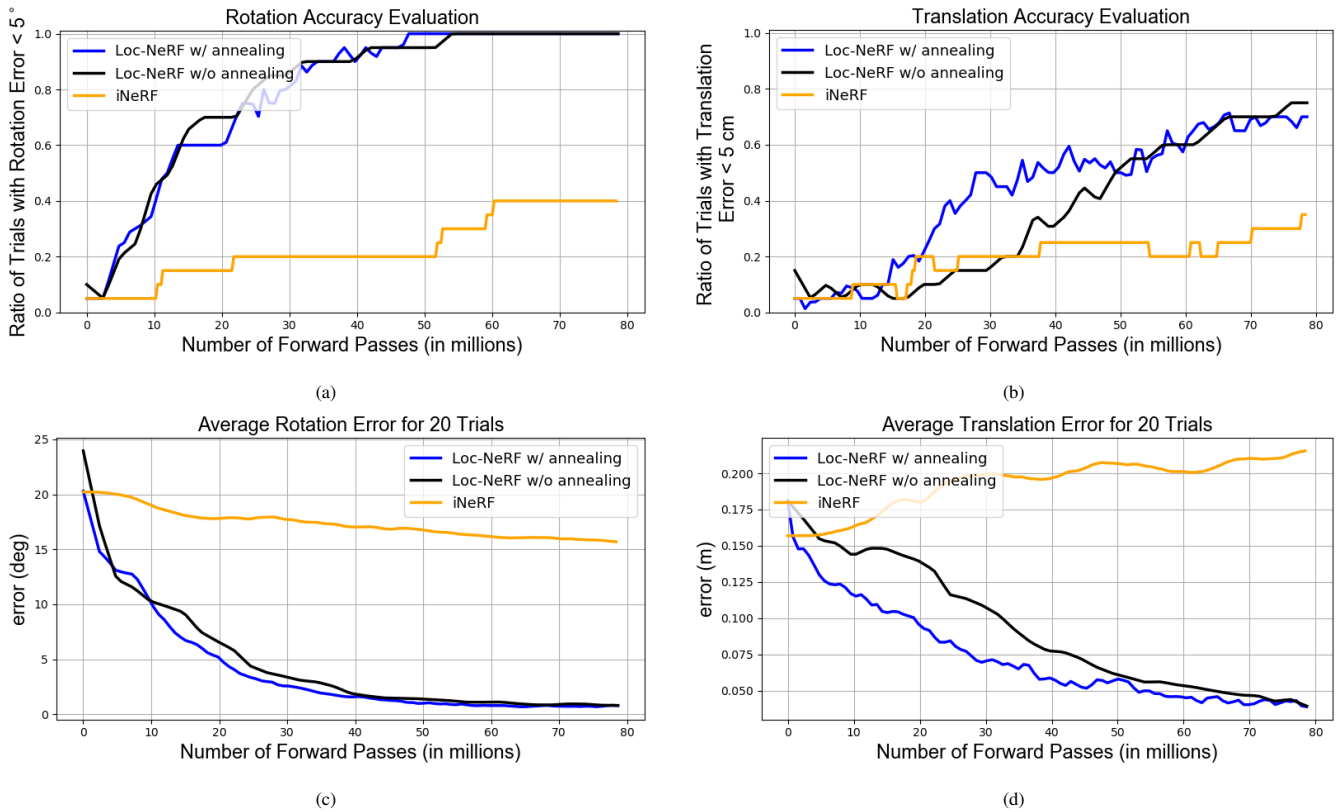


Fig. 2. Evaluation of Loc-NeRF and iNeRF on 20 camera poses from the LLFF dataset. As an ablation study of our annealing step, we also include results of Loc-NeRF without using Algorithm 1. (a) Ratio of trials with rotation error $< 5^\circ$. (b) Ratio of trials with translation error < 5 cm. (c) Average rotation error. (d) Average translation error.

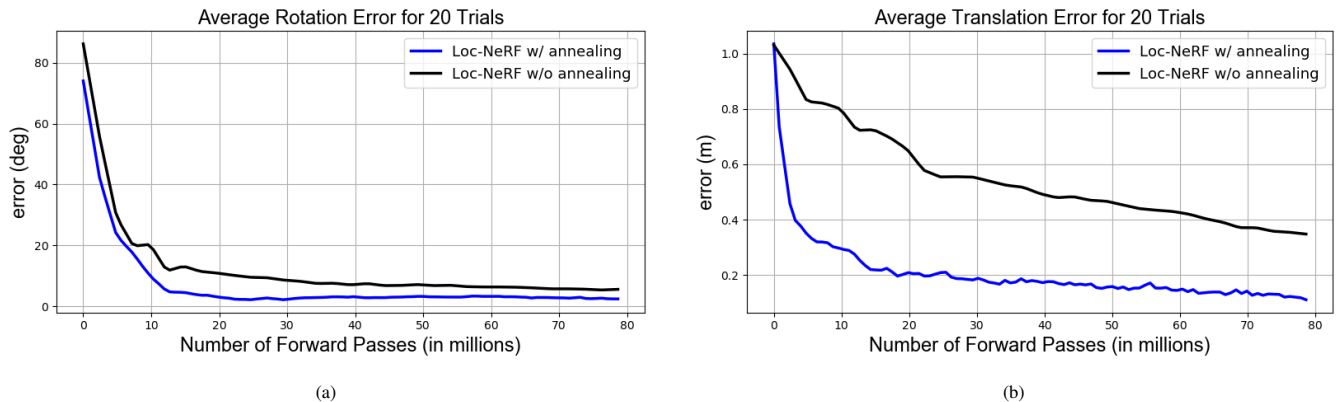


Fig. 3. Evaluation of Loc-NeRF on 20 camera poses from the LLFF dataset without an initial guess for the unknown pose. (a) Average rotation error. (b) Average translation error.

the starting pose of the drone is well known even though that is not a requirement for Loc-NeRF. The process loss of NeRF-Navigation gives it added robustness to portions of the trajectory where the NeRF rendering is of lower quality. However, Loc-NeRF is still able to achieve lower errors for both position and rotation on average and is able to recover from inaccurate pose estimates.

C. Full System Demonstration

Finally, we demonstrate our full system running in real-time on real data collected by a robot. We pre-train a NeRF model using NeRF-Pytorch [50] with metric scaled poses and

images from a Realsense d455 camera carried by a person. To run Loc-NeRF, we use a Realsense d455 as the vision sensor mounted on a Clearpath Jackal UGV. The prediction step for Loc-NeRF is performed using VINS-Fusion [40]. We log images and IMU data from the Jackal and then run VINS-Fusion and Loc-NeRF simultaneously on a laptop with an RTX 5000 GPU.

We initialize particles across a $1 \times 0.5 \times 3.5$ m area with a uniformly distributed yaw in $[-180^\circ, +180^\circ]$ and uniformly distributed roll and pitch in $[-2.5^\circ, +2.5^\circ]$ (again, the latter are directly observable from the IMU). Loc-NeRF receives data



Fig. 4. Example of NeRF rendering of a scene from Stonehenge.

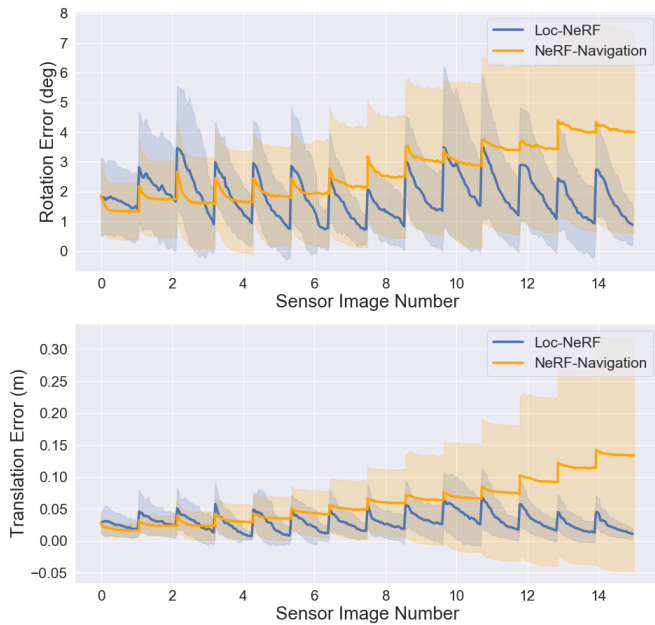


Fig. 5. Translation and rotation error of Loc-NeRF and NeRF-Navigation averaged over 18 trials. The shaded area shows one standard deviation above and below the mean error. The area between each sensor image number shows the optimization steps. Spikes at the beginning of each sawtooth show error when an image is first received and the pose is forward propagated with a dynamics model, and the bottom of each sawtooth represents the final pose estimate after optimization. For a fair comparison, both methods run the same number of forward passes for each camera image.

at the real-time rate. The prediction step runs at the nominal VIO rate of 15 Hz. Loc-NeRF starts with 400 particles which reduces to 150 during particle annealing. We set M to 32. With 400 particles the update step runs at approximately 0.9 Hz and then accelerates to 2.5 Hz with 150 particles during annealing. In this experiment, the particles quickly converge enough to trigger the annealing stage after about 6 update steps.

To qualitatively demonstrate that Loc-NeRF converges to the correct pose, we render a full image from NeRF using the pose estimated by Loc-NeRF and compare it with the corresponding camera image. We provide results from this

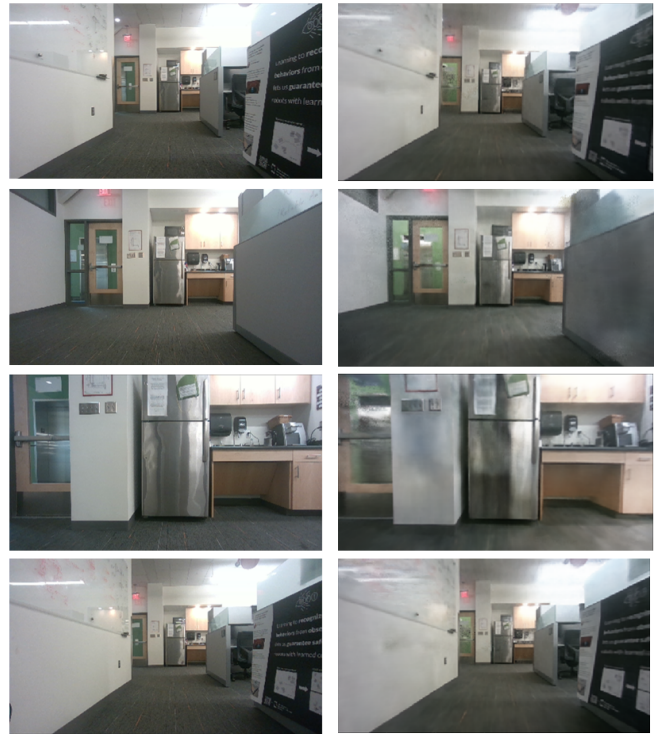


Fig. 6. Left Column: true images viewed by the camera. Right Column: NeRF-rendered images using the pose estimate from Loc-NeRF. Images correspond to update steps number 20, 40, 60, and 100 which occur at 13, 20, 28, and 44 seconds into the experiment, respectively.

test in Fig. 6 at selected points in the trajectory.

VI. CONCLUSION

We propose Loc-NeRF, a Monte Carlo localization approach that uses a Neural Radiance field (NeRF) as a map representation. We show how to incorporate NeRF in the update step of the filter, while the prediction step can be done using existing techniques (*e.g.*, visual-inertial navigation or by leveraging the robot dynamics). We show Loc-NeRF is the first approach to perform localization with NeRF from a poor initial guess, and can be used for global localization on small scale scenes. We have also demonstrated the ability to perform real-time localization with Loc-NeRF on data collected with a real-world robotic platform. Future work includes using adaptive techniques to adjust the number of particles [52] as well as scaling up localization to larger environments using bigger NeRF models such as [26] and [28]. Additionally, computation time can be reduced by leveraging recent work in faster NeRF rendering such as [51].

ACKNOWLEDGEMENT

The authors would like to gratefully acknowledge Timothy Chen, Michal Adamkiewicz, and all the authors of NeRF-Navigation who assisted us with benchmarking their work. We also acknowledge Jared Strader for assisting with collecting experimental data.

REFERENCES

- [1] P. Wang, X. Huang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The ApolloScape open dataset for autonomous driving and its application," *IEEE Trans. Pattern Anal. Machine Intell.*, 2019.
- [2] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "kpm: Keypoint affordances for category-level robotic manipulation," in *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, 2019.
- [3] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnnp: An accurate o (n) solution to the pnp problem," *Intl. J. of Computer Vision*, vol. 81, no. 2, p. 155, 2009.
- [4] L. Ke, S. Li, Y. Sun, Y.-W. Tai, and C.-K. Tang, "GSNet: joint vehicle pose and shape reconstruction with geometrical and scene-aware supervision," in *European Conf. on Computer Vision (ECCV)*. Springer, 2020, pp. 515–532.
- [5] X. Deng, J. Geng, T. Bretl, Y. Xiang, and D. Fox, "iCaps: iterative category-level object pose and shape estimation," *IEEE Robotics and Automation Letters*, 2022.
- [6] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "NICE-SLAM: Neural implicit scalable encoding for slam," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2022.
- [7] J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019.
- [8] 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 *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.
- [9] E. Sucar, S. Liu, J. Ortiz, and A. Davison, "iMAP: Implicit mapping and positioning in real-time," in *Intl. Conf. on Computer Vision (ICCV)*, 2021.
- [10] 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," *arXiv preprint arXiv:2003.08934*, 2020.
- [11] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [12] L. Ma, X. Li, J. Liao, Q. Zhang, X. Wang, J. Wang, and P. V. Sander, "Deblur-NeRF: Neural radiance fields from blurry images," *arXiv preprint arXiv:2111.14292*, 2021.
- [13] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-only robot navigation in a neural radiance world," *CoRR*, vol. abs/2110.00168, 2021. [Online]. Available: <https://arxiv.org/abs/2110.00168>
- [14] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo Localization for mobile robots," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 1999.
- [15] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler, "Neural geometric level of detail: Real-time rendering with implicit 3d shapes," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 11 358–11 367.
- [16] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems (NIPS)*, vol. 33, pp. 7537–7547, 2020.
- [17] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1366–1373.
- [18] V. Reijnders, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps," *IEEE Robotics and Automation Letters*, 2020.
- [19] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, "Kimeria: from SLAM to spatial perception with 3D dynamic scene graphs," *Intl. J. of Robotics Research*, vol. 40, no. 12–14, pp. 1510–1546, 2021, arXiv preprint: 2101.06894, (pdf).
- [20] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4460–4470.
- [21] S. Peng, C. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger, "Shape as points: A differentiable poisson solver," *Advances in Neural Information Processing Systems (NIPS)*, vol. 34, pp. 13 032–13 044, 2021.
- [22] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, "NeRF—: Neural radiance fields without known camera parameters," *arXiv preprint arXiv:2102.07064*, 2021.
- [23] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, "BARF: Bundle-adjusting neural radiance fields," in *Intl. Conf. on Computer Vision (ICCV)*, 2021.
- [24] Q. Meng, A. Chen, H. Luo, M. Wu, H. Su, L. Xu, X. He, and J. Yu, "GNeRF: GAN-based Neural Radiance Field without Posed Camera," in *Intl. Conf. on Computer Vision (ICCV)*, 2021.
- [25] J. Zhang, F. Zhan, R. Wu, Y. Yu, W. Zhang, B. Song, X. Zhang, and S. Lu, "VMRF: View matching neural radiance fields," 2022. [Online]. Available: <https://arxiv.org/abs/2207.02621>
- [26] H. Turki, D. Ramanan, and M. Satyanarayanan, "Mega-NeRF: Scalable construction of large-scale nerfs for virtual fly-throughs," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 12 922–12 931.
- [27] Y. Xiangli, L. Xu, X. Pan, N. Zhao, A. Rao, C. Theobalt, B. Dai, and D. Lin, "BungeeNeRF: Progressive neural radiance field for extreme multi-scale scene rendering," in *European Conf. on Computer Vision (ECCV)*, 2022.
- [28] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, "Block-NeRF: Scalable large scene neural view synthesis," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 8248–8258.
- [29] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "NeRF++: Analyzing and improving neural radiance fields," *arXiv:2010.07492*, 2020.
- [30] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-NeRF 360: Unbounded anti-aliased neural radiance fields," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [31] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022. [Online]. Available: <https://doi.org/10.1145/3528223.3530127>
- [32] T. Neff, P. Stadlbauer, M. Parger, A. Kurz, J. H. Mueller, C. R. A. Chaitanya, A. S. Kaplanyan, and M. Steinberger, "DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks," *Computer Graphics Forum*, vol. 40, no. 4, 2021. [Online]. Available: <https://doi.org/10.1111/cgf.14340>
- [33] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo," in *Intl. Conf. on Computer Vision (ICCV)*, 2021, pp. 14 124–14 133.
- [34] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, "Depth-supervised NeRF: Fewer views and faster training for free," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2022.
- [35] Y. Wei, S. Liu, Y. Rao, W. Zhao, J. Lu, and J. Zhou, "NerfingMVS: Guided optimization of neural radiance fields for indoor multi-view stereo," in *Intl. Conf. on Computer Vision (ICCV)*, 2021, pp. 5610–5619.
- [36] B. Roessle, J. T. Barron, B. Mildenhall, P. P. Srinivasan, and M. Nießner, "Dense depth priors for neural radiance fields from sparse input views," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 12 892–12 901.
- [37] R. Clark, "Volumetric bundle adjustment for online photorealistic scene capture," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 6124–6132.
- [38] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "PlenOctrees for real-time rendering of neural radiance fields," in *Intl. Conf. on Computer Vision (ICCV)*, 2021.
- [39] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234.
- [40] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors," *arXiv preprint: 1901.03638*, 2019.
- [41] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016, arXiv preprint: 1606.05830, (pdf).
- [42] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Intl. Conf. on Computer Vision (ICCV)*. IEEE, 2011, pp. 2320–2327.

- [43] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.
- [44] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, “iSDF: Real-time neural signed distance fields for robot perception,” in *Robotics: Science and Systems (RSS)*, 2022.
- [45] D. Fox, “KLD-Sampling: Adaptive particle filters,” in *Advances in Neural Information Processing Systems (NIPS)*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14. MIT Press, 2001. [Online]. Available: <https://proceedings.neurips.cc/paper/2001/file/c5b2cebf15b205503560c4e8e6d1ea78-Paper.pdf>
- [46] R. Hartley, J. Trumpf, Y. Dai, and H. Li, “Rotation averaging,” *IJCV*, vol. 103, no. 3, pp. 267–305, 2013.
- [47] J. Manton, “A globally convergent numerical algorithm for computing the centre of mass on compact lie groups,” in *ICARCV 2004 8th Control, Automation, Robotics and Vision Conference*, vol. 3, 2004, pp. 2211–2216 Vol. 3.
- [48] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, “Local light field fusion: Practical view synthesis with prescriptive sampling guidelines,” *ACM Transactions on Graphics (TOG)*, 2019.
- [49] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: <http://www.blender.org>
- [50] L. Yen-Chen, “NeRF-pytorch,” <https://github.com/yenchenlin/nerf-pytorch/>, 2020.
- [51] J. Tang, “Torch-ngp: a pytorch implementation of instant-ngp,” 2022, <https://github.com/ashawkey/torch-ngp>.
- [52] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, “Monte Carlo Localization – Efficient position estimation for mobile robots,” in *Proc. 16th AAAI National Conference on AI*, 1999.