2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3503-7770-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IROS58592.2024.10801527

NVINS: Robust Visual Inertial Navigation Fused with NeRF-augmented Camera Pose Regressor and Uncertainty Quantification

Juyeop Han[1], Lukas Lao Beyer[1], Guilherme V. Cavalheiro[1], and Sertac Karaman[1]

***Abstract*— In recent years, Neural Radiance Fields (NeRF) have emerged as a powerful tool for 3D reconstruction and novel view synthesis. However, the computational cost of NeRF rendering and degradation in quality due to the presence of artifacts pose significant challenges for its application in real-time and robust robotic tasks, especially on embedded systems. This paper introduces a novel framework that integrates NeRF-derived localization information with Visual-Inertial Odometry (VIO) to provide a robust solution for real-time robotic navigation. By training an absolute pose regression network with augmented image data rendered from a NeRF and quantifying its uncertainty, our approach effectively counters positional drift and enhances system reliability. We also establish a mathematically sound foundation for combining visual inertial navigation with camera localization neural networks, considering uncertainty under a Bayesian framework. Experimental validation in a photorealistic simulation environment demonstrates significant improvements in accuracy compared to a conventional VIO approach.**

I. Introduction

A Neural Radiance Field (NeRF) is a recent type of machine learning method that can learn continuous representations of a scene's color and density properties given a set of training images, which can allow for the synthesis of high quality novel views [1]. Their impressive initial results, despite limitations, have garnered the attention of researchers. For robotic applications, NeRFs and similar methods provide a conceptually flexible way to fuse camera measurements into a representation that does not suffer from the curse of dimensionality and synthesizes useful information, such as an environment's geometry.

Preliminary research has started exploring the applicability of these representations in perception tasks such as localization [2], [3], navigation [4], [5], [6] and SLAM [7], [8], [9], [10]. General NeRF improvements and variations were also suggested in order to address some of their main issues, such as their significant computational requirements [11], [12], [13], quality degradation under multi-scale data [14] and failures under a few-shot setting [15]. Nevertheless, many challenges remain for robotics applications due to limited computational resources in embedded systems and the occurrence of artifacts and reconstruction errors, all the more common in less controlled environments.

In this paper, we propose a new framework to extract information from an imperfect NeRF representation of an environment and integrate it into a visual-inertial odometry (VIO) solution using a rigorous mathematical foundation, while maintaining computational feasibility for onboard systems. To circumvent the computational demands of the NeRF's rendering process, we distill its localization information into a Convolutional Neural Network (CNN) trained offline on generating pairs of poses and images. Similarly to [3], the CNN estimates the absolute pose associated with an image, thus providing evidence against position drift, akin to a GPS sensor. In addition, we incorporate uncertainty quantification and outlier rejection strategies to allow for tight integration with a VIO pipeline. We evaluate methods such as Monte Carlo (MC) dropout [16] and deep ensembles [17] as options for uncertainty quantification and evaluate the resulting navigation performance for several trajectories simulated in a photo-realistic environment [18].

The main contributions of this research are summarized as below:

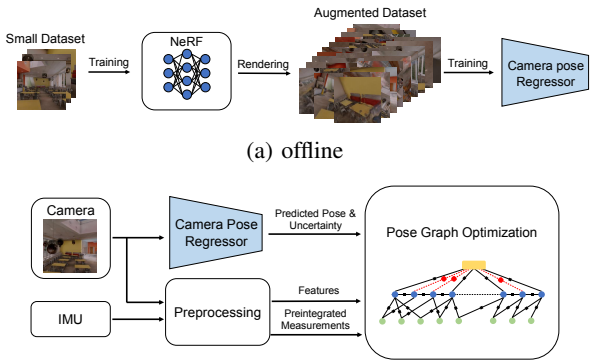
- We propose a real-time and loop closure-free VIO framework leveraging information extracted from an imperfectly trained NeRF that can be feasibly run on embedded hardware.
- We mathematically formulate the integration between VIO and the uncertain poses estimated by a neural network as *Maximum a Posteriori* (MAP) optimization in a Bayesian setting.
- We analyze the accuracy and efficiency improvements of the proposed framework compared to baselines.

II. Related Work

Robot Perception with Neural Scene Representation. Neural scene representations like NeRF have significantly advanced robotic perception capabilities, particularly in camera-only localization tasks. Techniques such as iNeRF [2] optimize camera poses by minimizing pixel residuals between real and NeRF-rendered images. Loc-NeRF [4] and LENS [3] enhance localization through particle filters and augmented data, respectively, while frameworks like that of Adamkiewicz *et al.* [5] integrate dynamics with NeRF for vision-only planning.

In dense SLAM, iMAP [8], NICE-SLAM [9] employ MLPs for pose estimation and mapping, albeit requiring depth data from RGB-D cameras. NeRF-SLAM [10] applies dense monocular SLAM to achieve faster and more precise scene representation compared to other methods. NeRF-VINS [6] tightly couples pre-trained NeRF with VIO by

[1]The authors are with Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139, USA {juyeop, llb, guivenca ,sertac}@mit.edu



(a) offline

(b) online

Fig. 1: Outline of the proposed VIO framework, NVINS: (a) In the offline phase, a NeRF is trained using a dataset gathered in the target environment. Novel views synthesized using this NeRF are used to train the camera pose regressor. (b) A pose predicted by the trained camera pose regressor, along with its associated uncertainty, is integrated with other sensor measurements via pose graph optimization.

comparing features from the real image and the image rendered by NeRF and performs in real-time on an onboard machine. However, the match between features of real and rendered images remains questionable in cases of low-resolution rendered images [19] or artifacts, often resulting from inadequate training, such as using sparse images.

CNN-based Camera Pose Regression and Uncertainty Quantification. CNNs are widely used for camera pose estimation from images, with PoseNet [20], [21], [22] being the pioneering CNN-based regressor. Research has explored structural enhancements like encoder-decoder [23] and LSTM architectures [24], and training improvements through prediction of relative camera poses [25].

Uncertainty in camera pose regression is captured through the covariance matrix of the pose, distinguishing between *aleatoric* and *epistemic uncertainty* [26]. Aleatoric uncertainty refers to the irreducible uncertainty associated with the noise inherent in the measurements. Epistemic uncertainty is the reducible uncertainty stemming from a lack of knowledge beyond the training dataset. Kendall *et al.* [21] quantify epistemic uncertainty using MC dropout to estimate errors caused by out-of-distribution inputs. To combine uncertainty estimated by the pose regressor with additional sensor measurements, conventional methods such as the Kalman filter [27] or visual odometry (VO) [28] have been used. Although epistemic and aleatoric uncertainty have been simultaneously estimated in prior work [28], we are not aware of any system that thoroughly utilizes this type of 6-DoF uncertainty estimate in a pose graph optimization framework.

III. Preliminaries

In this section, the formulation of two types of Bayesian Neural Networks (BNN) – deep ensemble and MC dropout – is presented. Note that deep ensemble can also be interpreted

(a) FlightGoggles (b) NeRF

Fig. 2: (a) Images captured in the Flightgoggles simulation environment [18]. (b) Images rendered through nerfstudio [29]. The images exhibit significant differences from the originals, primarily due to artifacts caused by the sparse density of training data.

from a Bayesian perspective [30], even though deep ensemble was initially described as a non-Bayesian technique [17].

Under the BNN framework, the neural network's weights, $\mathbf{w}$, are not considered fixed values. Instead, they initially follow a prior probabilistic distribution, $P(\mathbf{w})$. For a given training dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X}$ represents the input features and $\mathcal{Y}$ the target variable, these weights conform to a posterior distribution, $P(\mathbf{w}|\mathcal{D})$.

Deep Ensemble [17], [30]: In deep ensemble, a finite set of weights, $\mathcal{W}_p = \{\mathbf{w}_i^p\}$, is sampled from the prior distribution of weights, $P(\mathbf{w})$. Each sampled weight, $\mathbf{w}_i^p$, is optimized to find a maximum a posteriori (MAP) estimate $\mathbf{w}_i^* = \text{argmax}_{\mathbf{w}} P(\mathbf{w}|\mathcal{D}) = \text{argmin}_{\mathbf{w}}(-\log P(\mathcal{Y}|\mathbf{w}, \mathcal{X})P(\mathbf{w}))$ where $P(\mathcal{Y}|\mathbf{w}, \mathcal{X})$ is a likelihood of the label data $\mathcal{Y}$ given the input data $\mathcal{X}$ and weights $\mathbf{w}$. Then, the posterior distribution $P(\mathbf{w}|\mathcal{D})$ is approximated as

$$q(\mathbf{w}) \approx \frac{1}{|\mathcal{W}_p|} \sum_{i=1}^{|\mathcal{W}_p|} \delta(\mathbf{w} = \mathbf{w}_i^*) \tag{1}$$

where $\delta(\cdot)$ is a Dirac delta function.

The negative log-likelihood of the posterior is used to construct the loss function

$$\mathcal{L}_{total}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \mathcal{L}_{reg}(\mathbf{w}), \tag{2}$$

where the two components, $\mathcal{L}(\mathbf{w})$ and $\mathcal{L}_{reg}(\mathbf{w})$, correspond to the negative log-likelihood of $P(\mathcal{Y}|\mathbf{w}, \mathcal{X})$ and $P(\mathbf{w})$, respectively.

Monte Carlo (MC) dropout [16]: MC dropout is a variational inference method which approximates the true posterior distribution over weights $P(\mathbf{w}|\mathcal{D})$ as $q_{\theta^*}(\mathbf{w})$, a neural network with model parameters θ^*. At test time, weights are sampled from $q_{\theta^*}(\mathbf{w})$, i.e., $\hat{\mathbf{w}} = \{\hat{\mathbf{W}}_i\}_{i=1}^L \sim q_{\theta^*}(\mathbf{w})$. The weight matrix of i-th layer, $\hat{\mathbf{W}}_i$, is dropped out with a probability $1 - p_i$. That is, the weights at each layer i follow the Bernoulli distribution:

$$\mathbf{b}_j^i \sim \text{Bernoulli}(p_i) \quad \forall j \in \{1, \cdots, K_{i-1}\}, \tag{3}$$

$$\hat{\mathbf{W}}_i = \Theta_i \cdot \text{diag}(\mathbf{b}^i). \tag{4}$$

Here, K_i is the number of neurons at the layer i. To apply MC dropout at layer i, a vector of independent Bernoulli samples $\mathbf{b}^i$ is drawn. If j-th element of $\mathbf{b}^i$ is zero (i.e., $\mathbf{b}_j^i = 0$), the corresponding weight is dropped out.

We can find the optimal parameter θ^* minimizing the KL divergence $\mathrm{KL}(q_\theta(\mathbf{w})\|P(\mathbf{w}|\mathcal{D}))$, which directly corresponds to minimize $\mathcal{L}_{total}(\theta)$ with replacing weights $\mathbf{w}$ to the set of weight matrices, $\theta = \{\Theta_i\}_{i=1}^{L}$ [26].

IV. Training NeRF and Camera Pose Regressor with Uncertainty

In this section, NeRF is initially trained with a small dataset $\mathcal{D}_O = \{\mathcal{I}_O, \mathcal{T}_O\}$ to generate much larger dataset, $\mathcal{D}_A = \{\mathcal{I}_A, \mathcal{T}_A\}$, by rendering images corresponding to sampled poses using NeRF. Here, the dataset $\mathcal{D}_{\{O,A\}} = \{\mathcal{I}_{\{O,A\}}, \mathcal{T}_{\{O,A\}}\}$ consists of a set of images, $\mathcal{I}_{\{O,A\}}$, and their corresponding poses of camera, $\mathcal{T}_{\{O,A\}}$. Subsequently, the posterior of the camera pose $T_n \in \mathrm{SE}(3)$ given the dataset $\mathcal{D}_A$ and the input image I_n, $P(T_n|\mathcal{D}_A, I_n)$, is derived in the form of a Bayesian neural network (BNN) while quantifying its uncertainty. The output of the neural network and its uncertainty will be employed to compute the optimal state of VIO in Section V.

A. Training NeRF and Image Rendering

We use a NeRF reconstruction as a form of data augmentation, generating a large dataset $\mathcal{D}_A = \{\mathcal{I}_A, \mathcal{T}_A\}$ from a relatively small dataset $\mathcal{D}_O = \{\mathcal{I}_O, \mathcal{T}_O\}$. Even if the original poses $\mathcal{T}_O$ are not accurate, they can be refined using bundle adjustment techniques such as COLMAP [31]. A neural network $[\mathbf{c}, \sigma] = f_\phi(\mathbf{x}, \mathbf{d})$ parameterized by ϕ is trained with $\mathcal{D}_O$. The network takes two inputs: $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{d} \in \mathbb{R}^2$ representing the position and the direction of the ray for rendering, respectively. The outputs, $\mathbf{c}$ and σ, represent the view-dependent color and view-independent volume density of the sampled point of the ray. An intensity of one pixel is obtained through rendering a ray. The loss function of the neural network $f_\phi(\mathbf{x}, \mathbf{d})$ is designed to minimize the gap between the pixel intensity of the camera image and the estimated pixel intensity obtained through rendering process. For more information about NeRF, we refer to [1].

Once the NeRF reconstruction f_ϕ has been obtained, the augmented dataset $\mathcal{D}_A$ is constructed by rendering camera images $\mathcal{I}$ for corresponding camera poses $\mathcal{T}$. We obtain the set of poses $\mathcal{T}$ by sampling within a predefined area of interest. Note that, while we choose to use NeRF for data augmentation, other novel view synthesis techniques such as [11] and [12] can also be employed.

B. Camera Pose Regression with Uncertainty Quantification

In order to estimate a camera pose T_n given a query camera image I_n, we define the neural network $f_\mathbf{w} : \mathbb{R}^{W\times H\times 3} \to \mathbb{R}^{12}$ trained on the augmented dataset $\mathcal{D}_A = \{\mathcal{I}_A, \mathcal{T}_A\}$, where W and H are the width and height dimension of the image, and $\mathbf{w}$ is the weight of the neural network:

$$[\hat{\mathbf{t}}_n^T, (\boldsymbol{\sigma}_{n,p}^a)^T]^T = [f_\mathbf{w}^t(I_n), f_\mathbf{w}^\sigma(I_n)]^T = f_\mathbf{w}(I_n) \quad (5)$$

$I_n \in \mathbb{R}^{W\times H\times 3}$ is the input of the function $f_\mathbf{w}$. $\hat{\mathbf{t}}_n$ and $\boldsymbol{\sigma}_{n,p}^a$ are the vectorized representation of the pose and aleatoric uncertainty of the position, respectively. $f_\mathbf{w}$ utilizes a CNN to process high-dimensional image data. Notably, pixel coordinates are added as extra channels before each convolutional layer to improve CNN performance on coordinate transformation-related tasks [32]. The vectorized pose representation, $\hat{\mathbf{t}}_n = [\hat{\mathbf{p}}_n^T, \hat{\mathbf{r}}_n^T]^T \in \mathbb{R}^9$, consists of a 3D position and a 6D representation of rotation. This choice avoid the high errors associated with discontinuous representations such as axis-angle or quaternion [33]. We approximate the covariance of the position $\mathbf{p}_n$ as a diagonal matrix, i.e. $S_{n,p}^a = \mathrm{diag}(\boldsymbol{\sigma}_{n,p}^a)$.

We assume that the neural network (5), $f_\mathbf{w}$, models the distribution of estimated pose as a combination of Gaussian distribution of position and isotropic Langevin distribution of rotation given the weight $\mathbf{w}$ and query image I_n as an input:

$$P(\mathbf{t}_n|I_n, \mathbf{w}) = \frac{\exp\big(-\frac{1}{2}\|\mathbf{p}_n - \hat{\mathbf{p}}_n\|_{S_n^a}^2 + \kappa \mathrm{tr}(\hat{R}_n^T R_n)\big)}{c(\kappa)\sqrt{(2\pi)^3 \det(S_{n,p}^a)}} \quad (6)$$

with a rotation matrix $R_n \in \mathrm{SO}(3)$ and a normalizing factor, $c(\kappa)$. S_n^a and κ^{-1} are interpreted as an *aleatoric uncertainty*, an irreducible uncertainty caused by a random noise. Note that *epistemic uncertainty*, which will be represented later, has the same dimension as the aleatoric uncertainty to reflect the pose distribution (6). In this research, we treat the inverse of the rotational aleatoric uncertainty κ as a known constant due to the absence of an analytical form for $c(\kappa)$, which complicates the modeling of $c(\kappa)$ [34].

We leverage the BNN to model an *epistemic uncertainty*, which arises when the model encounters inputs that deviate from the training dataset $\mathcal{D}_A$. As shown in Fig. 2, rendered images $\mathcal{I}_\mathcal{A}$ cannot be perfectly identical to the original. When the model takes the images in the real environment, the gap between the rendered images and the original images is modeled as epistemic uncertainty. Unlike conventional neural networks, in a BNN, the weight $\mathbf{w}$ does not assume a fixed value but follows a posterior weight distribution $P(\mathbf{w}|\mathcal{D}_A)$ given the dataset $\mathcal{D}_A$, as discussed in Sec. III. To approximate the posterior weight distribution, we adopt deep ensemble [17] and MC dropout [16] due to their simplicity and scalability compared to other approaches [35].

The loss function for the weight $\mathbf{w}$ is formulated as shown in (2). Assuming independence between each data pair $(I_n, \mathbf{t}_n)$, the loss function $\mathcal{L}(\mathbf{w})$ corresponding to the negative log-likelihood of $P(\mathcal{T}_A|\mathcal{I}_A, \mathbf{w})$ is represented as

$$\begin{aligned}\mathcal{L}(\mathbf{w}) = &\frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}\Big(\kappa\|R_n - \hat{R}_n\|_F^2 + \\ &\|\mathbf{p}_n - \hat{\mathbf{p}}_n\|_{S_{n,p}^a}^2 + \mathrm{logdet}(S_{n,p}^a)\Big).\end{aligned} \quad (7)$$

For numerical stability, the covariance output of the network $\boldsymbol{\sigma}_n^a$ is replaced with the log vector of the covariance output, $\mathbf{s}_{n,p}^a$, such that $\sigma_{n,p,m}^a = \exp(s_{n,p,m}^a)$. Both the estimated mean, $\hat{\mathbf{t}}_n$, and log of variance, $\mathbf{s}_{n,p}^a$, are parameterized by $\mathbf{w}$. Additionally, the regularization loss, $\mathcal{L}_{reg}(\mathbf{w})$, assumes the prior distribution $P(\mathbf{w})$ is Gaussian with zero mean,

formulated as $\mathcal{L}_{reg}(\mathbf{w}) = \lambda\|\mathbf{w}\|_2^2$ with a positive constant λ.

A distribution of a camera pose, $\mathbf{t}_n$, given the input image, I_n, and training dataset $\mathcal{D}_A$ is derived by marginalizing over the weight of BNN:

$$P(\mathbf{t}_n|\mathcal{D}_A, I_n) = \int_{\mathbf{w}} P(\mathbf{t}_n|I_n, \mathbf{w})P(\mathbf{w}|\mathcal{D}_A)d\mathbf{w} \quad (8)$$

For practical implementation, a weight $\mathbf{w}_i$ is sampled from the distribution $q(\mathbf{w})$, approximating $P(\mathbf{w}|\mathcal{D}_A)$ through either deep ensemble or MC dropout. This results in a set of weights $\mathcal{W} = \{\mathbf{w}_i\}$. The pose distribution $P(\mathbf{t}_n|\mathcal{D}_A, I_n)$ is parameterized as the average and covariance of the outputs from neural networks with the sampled weights, represented as $f_{\mathbf{w}_i}(I_n) = [(\mathbf{t}_n^i)^T\ (\boldsymbol{\sigma}_{n,p}^{a,i})^T]^T,\ \forall \mathbf{w}_i \in \mathcal{W}$. The averages of the position and the orientation are estimated as:

$$\bar{\mathbf{p}}_n = \frac{1}{|\mathcal{W}|}\sum_{w_i\in\mathcal{W}} \mathbf{p}_n^i,\ \bar{R} = \underset{R\in \mathrm{SO(3)}}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{W}|} \|R_n^i - R\|_F^2 \quad (9)$$

The covariance, $\hat{\Sigma}_{n,\{p,r\}} = \hat{\Sigma}^a_{n,\{p,r\}} + \hat{\Sigma}^e_{n,\{p,r\}}$, of the pose distribution (8) is estimated as the sum of the aleatoric uncertainty, $\hat{\Sigma}^a_{n,\{p,r\}}$, and the epistemic uncertainty, $\hat{\Sigma}^e_{n,\{p,r\}}$ with $\{p,\ r\}$ representing position and rotation [26]. Assuming the distribution of $\|R_n - \bar{R}_n\|_F$ can be approximated by a zero-mean Gaussian distribution, the uncertainties can be represented as follows:

$$\begin{aligned} \hat{\Sigma}^a_{n,p} = \frac{\sum_{w_i\in\mathcal{W}} S_n^{a,i}}{|\mathcal{W}|-1}, \hat{\Sigma}^e_{n,p} = \frac{\sum_{w_i\in\mathcal{W}} \tilde{\mathbf{p}}_n^i(\tilde{\mathbf{p}}_n^i)^T}{|\mathcal{W}|-1} \\ \hat{\Sigma}^a_{n,r} = \kappa^{-1}, \hat{\Sigma}^e_{n,r} = \frac{\sum_{w_i\in\mathcal{W}} \|R_n^i - \bar{R}_n\|_F^2}{|\mathcal{W}|-1} \end{aligned} \quad (10)$$

with $\tilde{\mathbf{p}}_n^i = \mathbf{p}_n^i - \bar{\mathbf{p}}_n$. The estimated average, $\bar{T}_n \in \mathrm{SE(3)}$ corresponding $\bar{\mathbf{p}}_n$ and $\bar{R}$, and covariance, $\hat{\Sigma}_n$, of the pose will be utilized into the VIO framework to compute the residual function (16).

V. VIO Integration with Camera Pose Regressor

In this section, we describe how the BNN's absolute pose estimate is incorporated into a visual-inertial odometry (VIO) framework. We view this problem as maximum a posteriori (MAP) inference on a factor graph. In particular, we aim to acquire the optimal estimate of the state $\mathcal{X} = \{\mathcal{X}_B,\ \mathbf{P}\}$:

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}}\, P(\mathcal{X}|\mathcal{D}_A, \mathcal{Z}), \quad (11)$$

through factor graph optimization, where

$$\mathcal{X}_B = \{\mathbf{x}_B^{(1)}, \cdots, \mathbf{x}_B^{(N)}\}, \quad \mathbf{P} = [\mathbf{p}_1 \cdots \mathbf{p}_M]^T, \quad (12)$$

$$\mathcal{D}_A = \{\mathcal{I}_A, \mathcal{T}_A\}, \text{and} \quad \mathcal{Z} = \{\mathcal{Z}_B, \mathcal{Z}_C, \mathcal{I}_o\}. \quad (13)$$

Here, N and M denote the number of poses in the discretized trajectory estimate and the number of observed landmarks, respectively. Each pose $\mathbf{x}_B^{(n)} \in \mathrm{SE(3)}$ describes the world-to-body transform at time t_n. The collection of observed landmarks is denoted by $\mathbf{P} \in \mathbb{R}^{M\times 3}$. The augmented training dataset introduced in Section IV and the sensor inputs (composed of IMU measurements $\mathcal{Z}_B$, tracked features extracted by a VIO frontend $\mathcal{Z}_C$, and camera images $\mathcal{I}_o$) are denoted by $\mathcal{D}_A$ and $\mathcal{Z}$, respectively. We model the IMU according to [36], and have omitted details such as velocity and bias walk estimation of IMU for notational simplicity.

Moreover, we selectively exclude the neural network output identified as outliers from the optimization, ensuring a more robust MAP estimation using the uncertainty of the neural network.

A. Factor Graph Optimization

Assuming zero-mean Gaussian noise for all measurements and that all measurements are independent of each other except the relation between the offline data $\mathcal{D}_A$ and camera images $\mathcal{I}_o$, the MAP estimate (11) is given by

$$\begin{aligned} \mathcal{X}^* &= \underset{\mathcal{X}}{\operatorname{argmax}} P(\mathcal{X}|\mathcal{D}_A, \mathcal{I}_o)P(\mathcal{Z}_B, \mathcal{Z}_C|\mathcal{X}, \mathcal{D}_A, \mathcal{I}_o) \\ &= \underset{\mathcal{X}}{\operatorname{argmax}} P(\mathcal{X}|\mathcal{D}_A, \mathcal{I}_o)P(\mathcal{Z}_B|\mathcal{X})P(\mathcal{Z}_C|\mathcal{X}) \\ &= \underset{\mathcal{X}}{\operatorname{argmax}} \Big\{ \prod_{n\in\mathcal{N}} P(T_n|\mathcal{D}_A, I_n) \\ &\qquad \prod_{b\in\mathcal{B}} P(\mathbf{z}_{b_{k+1}}^{b_k}|\mathcal{X}) \prod_{(i,j)\in\mathcal{C}} P(\mathbf{z}_i^j|\mathcal{X}) \Big\}. \end{aligned} \quad (14)$$

The posterior camera pose $P(T_n|\mathcal{D}_A, I_n)$ is given by the output of the BNN (8). $P(\mathbf{z}_{b_{k+1}}^{b_k}|\mathcal{X})$ and $P(\mathbf{z}_i^j|\mathcal{X})$ are likelihoods of IMU and image feature measurements, where $\mathbf{z}_i^j \in \mathbb{R}^2$ refers to the i-th feature captured by the j-th camera frame. Note that we use IMU preintegration [36], so that $\mathbf{z}_{b_{k+1}}^{b_k}$ corresponds to the integrated IMU measurement in the time interval $[t_k, t_{k+1})$.

The nonlinear optimization problem (14) is solved by minimizing the sum of Mahalanobis distances:

$$\begin{aligned} \mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmin}} \Big\{ &\sum_{n\in\mathcal{N}} \|\mathbf{r}_{\mathcal{N}}(I_n, \mathcal{X})\|^2_{\hat{\Sigma}_n} \\ &+ \sum_{k\in\mathcal{B}} \|\mathbf{r}_{\mathcal{B}}(\mathbf{z}_{b_{k+1}}^{b_k}, \mathcal{X})\|^2_{\Sigma_{b_{k+1}}^{b_k}} + \sum_{(i,j)\in\mathcal{C}} \|\mathbf{r}_{\mathcal{C}}(\mathbf{z}_i^j, \mathcal{X})\|^2_{\Sigma_i^j} \Big\} \end{aligned} \quad (15)$$

The residual $\mathbf{r}_{\mathcal{N}}(I_n, \mathcal{X})$ is defined as the combination of the Euclidean position error and the Frobenius norm of the rotational error, $\bar{T}_n$, obtained from the BNN, and the actual camera pose, T_n:

$$\mathbf{r}_{\mathcal{N}}(I_n, \mathcal{X}) = [(\mathbf{p}_n - \bar{\mathbf{p}}_n)^T,\ \|R_n - \bar{R}_n\|_F]^T. \quad (16)$$

For convenience of implementation, $\|R_n - \bar{R}_n\|_F$ in (16) is replaced with $\sqrt{2}\|\mathrm{Log}(\bar{R}_n^T R_n)\|_2$, since

$$\|\mathrm{Log}(\bar{R}_n^T R_n)\|_2 = \epsilon,\ \|R_n - \bar{R}_n\|_F = 2\sqrt{2}\sin\frac{\epsilon}{2} \approx \sqrt{2}\epsilon, \quad (17)$$

where ϵ is the angle between the two rotations, and $\mathrm{Log}(\cdot)$ refers to the logarithm map on SO(3). $\mathbf{r}_{\mathcal{B}}(\mathbf{z}_{b_k}^{bk+1}, \mathcal{X})$ and $\mathbf{r}_{\mathcal{C}}(\mathbf{z}_i^j, \mathcal{X})$ represent the residuals based on IMU measurements [36] and the camera feature measurements [37], respectively.

B. Uncertainty-based Outlier Rejection

As shown in Fig. 2, areas with little or no coverage in the original, unaugmented dataset $\mathcal{D}_{\text{NeRF}}$ cannot be reconstructed accurately. We find that the pose and uncertainty $\hat{\Sigma}_{n,(\cdot)}$ predicted by our BNN can be unreliable in such cases. Thus, we propose two simple and effective rejection heuristics to detect and discard pose regressor outputs that are deemed unreliable:

$$\text{tr}(\hat{\Sigma}_{n,p}) > \tau_1 \quad \text{or} \quad \|\mathbf{p}_n^* - \hat{\mathbf{p}}_n\|_{\hat{\Sigma}_{n,p}} > \tau_2. \tag{18}$$

First, we exclude predictions for which the magnitude of the trace of the positional uncertainty exceeds a certain threshold τ_1. Additionally, we compute the Mahalanobis distance between the most recent pose estimated by the VIO system and the current pose predicted by the pose regressor, under the covariance as predicted by the pose regressor. When this distance exceeds a predefined threshold τ_2, the BNN output is discarded.

VI. Experiments

Experiments are conducted to validate (a) improved pose accuracy achieved through increased data with NeRF and uncertainty quantification, (b) consistent pose estimation without the need for loop closure, and (c) computational efficiency of the proposed VIO framework. All experiments are conducted within the FlightGoggles simulation [18].

We compare the accuracy of the camera pose regressors trained with and without uncertainty prediction. We also assess the computational efficiency of the designed pose regressors on both a desktop and a Jetson AGX Xavier onboard machine (Sec. VI-A). The accuracy of our proposed VIO framework is then measured under different settings regarding uncertainties (Sec. VI-B).

We collected 1,297 posed 480 px×480 px RGB-D images within a 15m×24m×4m region of the "Stata Center" model available in the Flightgoggles simulation [18]. This dataset is used to train the *depth-nerfacto* model [29], which incorporates the multiresolution hash encoding from Instant-NGP [13] for improved performance and can make use of depth information.

Upon completing the training of the NeRF model, 50,000 pairs of poses and images were generated by the trained NeRF model and utilized to train the camera pose regressors. The orientations of the sampled poses are constrained in order to avoid extreme attitudes exceeding pitch or roll angles of approximately 30 degrees. The camera pose regressor architecture includes 4 CoordConv layers [32] followed by 2 (resp. 3) fully connected layers for the pose (resp. uncertainty) output. We constructed a relatively shallow network compared to other pose regressors [20], [25], [38] to enable real-time onboard inference. Training of the pose regressor networks was conducted using the Adam optimizer with a learning rate of 10^{-3} for 100 epochs. The weighting factor κ (see (7)) was set to 1. The weight decay factor for all networks is set to $\lambda = 10^{-5}$. These parameters were selected through trial and error to maximize the performance of the networks.

TABLE I: Inference speeds of camera pose regressors on desktop and Jetson AGX Xavier (ms)

Networks	Desktop	Jetson AGX Xavier
no-nerf	1.638 ± 0.122	8.378 ± 1.951
vanilla	1.603 ± 0.063	8.299 ± 0.866
dropout	13.989 ± 0.691	41.27 ± 1.949
ensemble	8.255 ± 0.166	29.91 ± 1.247

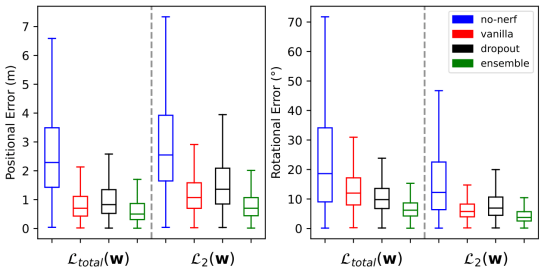


Fig. 3: Positional and rotational errors of various camera pose regressors trained with different loss functions across entire trajectories (m/degree)

A. Camera Pose Regressor with Uncertainty

Setup. In this subsection, we aim to validate (a) the improved performance through the increased number of training data aided by NeRF and adoption MC dropout and deep ensemble, (b) the effect of aleatoric uncertainty output $\boldsymbol{\sigma}_{n,p}^a$, and (c) computation speed of the camera pose regressor across different computation environment.

We trained pose regressors with two distinct loss functions: $\mathcal{L}_{total}(\mathbf{w})$ as defined in (2) and $\mathcal{L}_2(\mathbf{w})$ which is identical to $\mathcal{L}_{total}(\mathbf{w})$ except that the positional covariance $S_{n,p}^a$ of $\mathcal{L}_2(\mathbf{w})$ is considered constant. For each loss function, four camera pose regressors were trained and named as follows: **no-nerf**, **vanilla**, **dropout**, and **ensemble**. 'No-nerf' and 'vanilla' are neural networks trained with the dataset used to train the NeRF model and the dataset rendered by NeRF, respectively. 'Dropout' and 'ensemble' are neural networks designed to quantify the uncertainties of the outputs, utilizing MC dropout and deep ensemble, respectively. For the dropout implementation, the dropout rate was uniformly set at $p = 0.05$ for all layers. 'Ensemble' has 8 independent neural networks trained, each with their parameters initialized according to a Gaussian distribution with a standard deviation of $\alpha = 0.02$. To ensure a fair comparison, 'dropout' had 8 forward passes for inference.

After training, the pose regressors predict the camera pose attached to a quadrotor across seven different trajectories for approximately two minutes within the simulation environment. One trajectory follows the Lemniscate of Bernoulli (**Trajectory 1**), while the others are Lissajous curves, each with distinct parameters (**Trajectory 2 - 7**).

Results. Firstly, the inference speeds of the camera pose regressors, tested on 'Trajectory 1', are compared on a desktop equipped with Intel Core i9-7920X CPU at 2.90GHz and NVIDIA TITAN V 12GB GPU, and on an NVIDIA Jetson

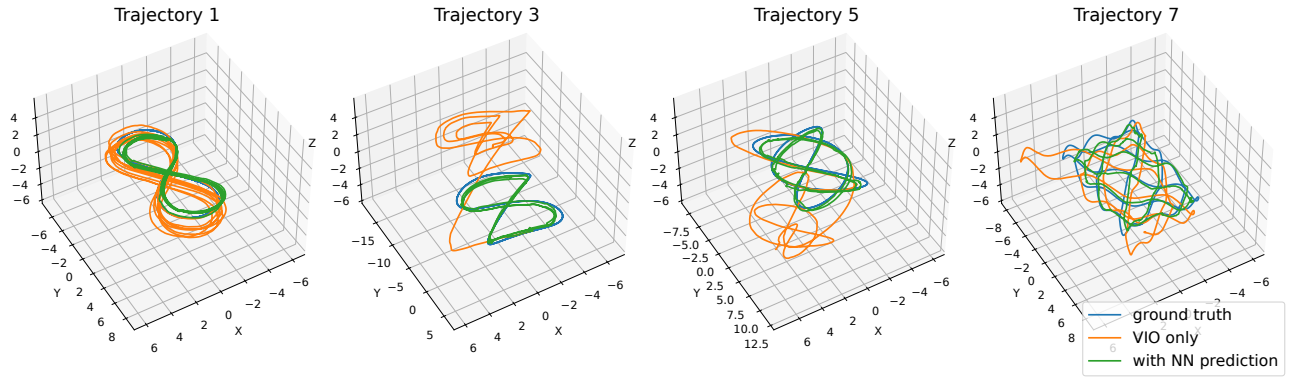


Fig. 4: Comparison of ground truth trajectories, 'Trajectory 1, 3, 5, and 7', (blue) with estimations from the VIO framework without the camera pose regressor (orange), and with the ensemble camera pose regressor augmented by uncertainty quantification and outlier rejection (green)

AGX Xavier, as shown in Table. I. Although the 'dropout' and 'ensemble' demonstrate inference speeds a few times slower than those of 'no-nerf' and 'vanilla', they are still sufficiently fast for VIO integration in real-time considering that the slowest inference speed, observed with 'dropout' on the Jetson AGX Javier, achieves about 24Hz.

Fig. 3 displays the errors for the trained pose regressors having loss functions $\mathcal{L}_2(\mathbf{w})$ and $\mathcal{L}_{total}(\mathbf{w})$, assessed across the whole testing trajectories. It is evident that generating training data via NeRF significantly increases the accuracy of the pose regressors. Furthermore, it is shown that pose regressors trained with $\mathcal{L}_{total}(\mathbf{w})$ exhibit better positional accuracy than those with $\mathcal{L}_2(\mathbf{w})$. On the other hand, pose regressors trained with $\mathcal{L}_{total}(\mathbf{w})$ demonstrate worse rotational accuracy. It is attributed to the neural network's uncertainty output, automatically weighing losses for position and rotation in $\mathcal{L}_{total}(\mathbf{w})$. Additionally, 'ensemble' consistently outperforms the other methods, contrary to 'dropout' which underperforms relative to 'vanilla' in terms of positional accuracy. The performance of 'dropout' might be improved by applying dropout to fewer layers, as suggested in [21].

B. Integration VIO with Camera Pose Regressor

Setup. Each trained pose regressor provided the VIO framework with a sequence of estimated camera poses and their uncertainties, $\{\bar{T}_n, \hat{\Sigma}_n\}_{n\in\mathcal{N}}$ used to compute the sum of residual norms of the residual functions (15, 16). We employed 'dropout' and 'ensemble' regressors trained with $\mathcal{L}_{total}(\mathbf{w})$.

To evaluate the effectiveness of incorporating the camera pose regressor into the VIO framework—specifically, to examine the impact of including uncertainty predictions on noise and deviation from the training dataset—we conducted tests across the seven trajectories described in the preceding subsection. Additionally, we established a baseline by testing the VIO framework without the camera pose regressor, referred to as **only VIO**. These tests were conducted under eight different scenarios, based on three criteria, named as **(option 1) + (option 2) + (option 3)**: *(option 1)* type of pose regressors ('dropout' or 'ensemble'), *(option 2)* the application of predicted uncertainty $\hat{\Sigma}_n$ within the VIO framework ('constant' or 'estimated'), and *(option 3)* the application of outlier rejection ('no rejection' and 'rejection'). Note that loop closure is not applied in any of the cases.

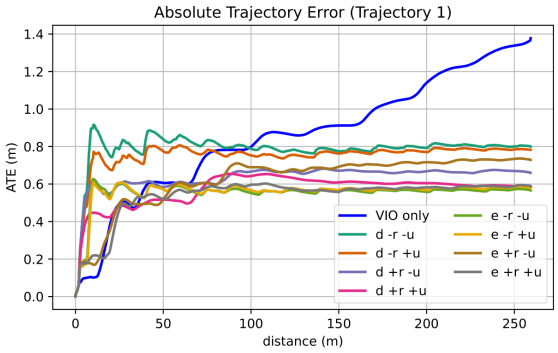


Fig. 5: Absolute Trajectory Error (ATE) for VIO with and without camera pose regressors over the travel distance of 'Trajectory 1'. 'd' (resp. 'e') denotes 'dropout' (resp. 'ensemble'), '+u' (resp. '+r') indicates uncertainty prediction (resp. outlier rejection) is applied, and '-' signifies the absence of the respective feature.

When 'constant' and 'rejection' are concurrently applied, only the Mahalanobis norm-based rejection is implemented, as the trace remains unchanged. In the 'constant' scenario, a fixed covariance, $\text{diag}([0.2; 0.2; 0.2; 1.0]^T)$, is utilized instead of $\hat{\Sigma}_n$. The thresholds for outlier rejection (18) are set at $\tau_1 = 1.0$ and $\tau_2 = 0.4$.

For the VIO pipeline, features are detected using GFTT [39] and tracked using the Lucas-Kanade algorithm [40]. The backend, using GTSAM [41], optimizes factor graphs at 17 Hz, combining features, 240 Hz IMU data, and pose regressor estimates. Pose regressor estimates are incorporated at every alternate keyframe, except when rejected based on specific heuristics (18).

Results. Table II presents the mean positional and rotational errors for each evaluated scenario across all trajectories. Figure 4 illustrates the actual and estimated trajectories for the 'only VIO' and 'ensemble + estimated + rejection'

TABLE II: Average positional and rotational errors for different VIO configurations over all test trajectories (m/degrees). The blue and red bold numbers indicate the first and second-best performances, respectively, for each test case, excluding the 'only VIO' configuration.

Trajectory	only VIO	constant + no rejection		estimated + no rejection		constant + rejection		estimated + rejection	
		dropout	ensemble	dropout	ensemble	dropout	ensemble	dropout	ensemble
1	1.224/1.01	0.659/5.76	**0.468**/2.43	0.659/6.14	0.472/2.88	0.942/1.44	0.601/**1.40**	0.470/**1.35**	**0.418**/1.51
2	4.104/3.13	1.225/6.80	0.771/5.28	1.205/12.43	0.762/4.53	0.942/**1.86**	**0.601**/2.92	0.882/**1.92**	**0.524**/1.99
3	11.917/4.58	1.201/6.16	0.750/4.22	1.241/6.40	0.724/3.61	0.947/1.98	**0.601**/2.24	0.760/**1.72**	**0.510**/**1.97**
4	1.411/1.80	1.221/8.88	0.793/5.19	1.069/7.70	0.746/5.26	0.999/5.02	**0.678**/**2.39**	**0.720**/**1.75**	0.815/5.39
5	6.476/2.04	0.918/7.17	0.543/3.13	0.802/10.73	0.548/3.79	0.846/2.55	**0.484**/**1.91**	0.512/2.05	**0.455**/**1.90**
6	6.876/2.12	0.902/7.37	**0.586**/4.43	0.806/6.21	**0.557**/4.47	0.950/**3.23**	0.660/**2.98**	0.686/4.50	0.586/4.60
7	1.767/1.35	0.836/4.56	0.567/3.29	0.715/5.30	**0.513**/3.35	0.786/1.95	**0.426**/1.97	0.724/2.10	0.553/2.42
Average	4.824/2.29	0.995/6.67	0.640/3.99	0.928/7.84	0.617/3.98	0.869/2.58	**0.592**/**2.26**	0.679/**2.20**	**0.552**/2.82

scenarios, specifically for Trajectories 1, 3, 5, and 7. Figure 5 displays the Absolute Trajectory Errors (ATE) for all scenarios along 'Trajectory 1'.

Fig. 4 and 5 demonstrate that a drift accumulates in the 'only VIO' scenario, resulting in an increase in positional error. On the other hand, when the VIO is integrated with the camera pose regressors, errors tend to converge, as the camera pose regressors effectively constrain the error bounds. The effectiveness of the pose regressors is evident in Table II, where the positional error for the 'only VIO' case is significantly higher than in other cases.

Table II represent that 'ensemble', 'estimated', and 'rejection' exhibit better positional accuracy compared 'dropout', 'constant', and 'no rejection', respectively. Specifically, the 'ensemble + estimated + rejection' scenario achieves the lowest average positional error , showing a 44.5% improvement in accuracy over the 'dropout + constant + no rejection' scenario which displays the highest positional error. Fig. 5 further illustrates that 'ensemble + estimated + rejection' configuration not only converges the lowest error along with 'dropout + estimated + rejection', 'ensemble + constant + no rejection', and 'ensemble + estimated + no rejection' configurations but also maintains a lower error than the configurations prior to convergence. As a result, the uncertainty quantification of the camera pose regressor and its integration into pose graph optimization, both as a covariance measure or through outlier rejection, enhances the overall positional performance.

Rotational errors are lower when 'rejection' configurations are applied, compared to 'no rejection', which is noteworthy as it suggests that position-based rejection also enhances rotational performance. This is because there exists a correlation between positional and rotational uncertainties, as indicated in [21]. Furthermore, 'rejection' scenarios exhibit rotational accuracies that are relatively comparable to the 'Only VIO' scenario, while other configurations tend to underperform relative to 'Only VIO' with respect to rotation. However, 'estimated' scenarios do not consistently outperform the 'constant' scenarios in terms of rotational accuracy.

In summary, the experiments demonstrate that integrating the camera pose regressor into the VIO framework prevents drift. Moreover, applying covariance and outlier rejections, derived from uncertainty quantification, improves the positional accuracy. Additionally, outlier rejection safeguards against the potential decline in rotational accuracy that may result from using the pose regressor.

VII. Conclusions

In this research, we introduced a framework that capitalizes on the capabilities of Neural Radiance Fields (NeRF) to enhance real-time and robust localization for robotic navigation by integrating a camera pose regressor, trained on augmented data generated from NeRF, with a VIO framework. The uncertainty of the pose regressor was quantified in a Bayesian manner, estimating the potential error between the real and estimated poses caused by discrepancies between the actual scene and its NeRF representation. This quantification was integrated into the VIO framework, enhancing its reliability and robustness. Our experimental validation with Flightgoggle simulation confirms that incorporating NeRF-aided camera pose regression, alongside uncertainty quantification, substantially improves the VIO framework's performance. Future research will explore the scalability of our method to larger environments, such as entire indoor floors, and investigate effectiveness of uncertainty quantification given the environmental changes over time after the NeRF training.

References

[1] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[2] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting neural radiance fields for pose estimation. In *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[3] Arthur Moreau, Nathan Piasco, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. LENS: Localization enhanced by nerf synthesis. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, pages 1347–1356, 2022.

[4] Dominic Maggio, Marcus Abate, Jingnan Shi, Courtney Mario, and Luca Carlone. Loc-NeRF: Monte carlo localization using neural radiance fields. In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4018–4025, 2023.

[5] 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*, 7(2):4606–4613, 2022.

[6] Saimouli Katragadda, Woosik Lee, Yuxiang Peng, Patrick Geneva, Chuchu Chen, Chao Guo, Mingyang Li, and Guoquan Huang. NeRF-VINS: A real-time neural radiance field map-based visual-inertial navigation system, 2023. arXiv:2309.09295.

[7] Louis Wiesmann, Tiziano Guadagnino, Ignacio Vizzo, Nicky Zimmerman, Yue Pan, Haofei Kuang, Jens Behley, and Cyrill Stachniss. Locndf: Neural distance field mapping for robot localization. *IEEE Robotics and Automation Letters*, 8(8):4999–5006, 2023.

[8] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6229–6238, October 2021.

[9] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. NICE-SLAM: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[10] Antoni Rosinol, John J. Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. In *Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3437–3444, 2023.

[11] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5501–5510, June 2022.

[12] 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.

[13] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4), July 2022.

[14] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5855–5864, October 2021.

[15] Jiawei Yang, Marco Pavone, and Yue Wang. FreeNeRF: Improving few-shot neural rendering with free frequency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8254–8263, 2023.

[16] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[17] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 30, 2017.

[18] Winter Guerra, Ezra Tal, Varun Murali, Gilhyun Ryou, and Sertac Karaman. FlightGoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6941–6948, 2019.

[19] Fang Zhu, Shuai Guo, Li Song, Ke Xu, and Jiayu Hu. Deep review and analysis of recent nerfs. *APSIPA Transactions on Signal and Information Processing*, 12(1):–, 2023.

[20] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[21] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4762–4769, 2016.

[22] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[23] Iaroslav Melekhov, Juha Ylioinas, Juho Kannala, and Esa Rahtu. Image-based localization using hourglass networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[24] Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using lstms for structured feature correlation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[25] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-aware learning of maps for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[26] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Proceedings of the Advances in Neural Information Processing Systems*, volume 30, 2017.

[27] Arthur Moreau, Nathan Piasco, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. Coordinet: Uncertainty-aware pose regressor for reliable vehicle localization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2229–2238, January 2022.

[28] Valentin Peretroukhin, Brandon Wagstaff, Matthew Giamou, and Jonathan Kelly. Probabilistic regression of rotations using quaternion averaging and a deep multi-headed network, 2020. arXiv:1904.03182.

[29] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023.

[30] Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Advances in Neural Information Processing Systems*, volume 33, pages 4697–4708, 2020.

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

[32] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 31, 2018.

[33] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[34] Alessandro Chiuso, Giorgio Picci, and Stefano Soatto. Wide-sense estimation on the special orthogonal group. *Communications in Information and Systems*, 8(3):185 – 200, 2008.

[35] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

[36] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2017.

[37] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment — a modern synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms: Theory and Practice*, pages 298–372, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[38] Arthur Moreau, Nathan Piasco, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. Coordinet: Uncertainty-aware pose regressor for reliable vehicle localization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2229–2238, January 2022.

[39] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[40] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981.

[41] Frank Dellaert and GTSAM Contributors. `https://github.com/borglab/gtsam`, May 2022.