Research Article.

NeRF-VIO: Map-Based Visual-Inertial Odometry with Initialization Leveraging Neural Radiance Fields

Yanyu Zhang[1], Dongming Wang[1], Jie Xu[1], Mengyuan Liu[2], Pengxiang Zhu[1], Wei Ren[1]

1. Department of Electrical and Computer Engineering, University of California, Riverside, United States; 2. Northeastern University, United States

A prior map serves as a foundational reference for localization in context-aware applications such as augmented reality (AR). Providing valuable contextual information about the environment, the prior map is a vital tool for mitigating drift. In this paper, we propose a map-based visual-inertial localization algorithm (NeRF-VIO) with initialization using neural radiance fields (NeRF). Our algorithm utilizes a multilayer perceptron model and redefines the loss function as the geodesic distance on $SE(3)$, ensuring the invariance of the initialization model under a frame change within $\mathfrak{se}(3)$. The evaluation demonstrates that our model outperforms existing NeRF-based initialization solution in both accuracy and efficiency. By integrating a two-stage update mechanism within a multi-state constraint Kalman filter (MSCKF) framework, the state of NeRF-VIO is constrained by both captured images from an onboard camera and rendered images from a pre-trained NeRF model. The proposed algorithm is validated using a real-world AR dataset, the results indicate that our two-stage update pipeline outperforms MSCKF across all data sequences.

I. Introduction and Related Work

Augmented Reality (AR) [1][2] and Virtual Reality (VR) [3][4] have emerged as transformative technologies, offering immersive experiences across various domains. One critical aspect shaping the effectiveness of these experiences is the incorporation of prior maps [5]. These maps provide essential spatial context, enabling accurate localization, tracking, and seamless integration of virtual elements into the real world. To achieve high quality and low latency user experiments, visual-inertial navigation systems (VINS) have

 doi.org/10.32388/VAM5O9

received considerable popularity in AR/VR applications [6][7][8][9][10] through utilizing low-cost and lightweight onboard cameras and inertial measurement units (IMUs).

Using VINS, the drift of the pose will accumulate and the uncertainty of the estimate will grow unbounded without global information, such as a prior map, GNSS measurement, or loop closure. However, GNSS may not be applicable indoors, and loop closure demands both a precise and efficient place recognition algorithm [11][12][13] and substantial memory space to store historical features [14][15]. Consequently, prior map-based approaches have gained significant interest over the past few decades [16][17][18][19][20][21].

One of the key challenges that the map-based VINS literature tackles is relocalization based on one image and a prior map. Typically, descriptor-based methods are employed to establish 2D-3D correspondences by reprojecting map points to the image frame and matching them with features extracted from the image[22][23]. Considering the increase in optimization complexity with map size, DBoW[11] represents an image by the statistic of different kinds of features from a visual vocabulary. Inspired by the DBoW, keyframe-based loop closure detection and localization are employed in ORB-SLAM[14] and ORB-SLAM2[15]. However, DBoW sacrifices spatial information about features, potentially leading to ambiguities or inaccuracies.

Recently, Neural radiance fields (NeRF)[24] introduces a multilayer perceptron (MLP) to capture a radiance field representation of a scene. During training, NeRF estimates the color and density of sampled particles along each ray, and minimizes the photometric error between the estimated image and the groundtruth. NICE-SLAM[25] proposes a dense simultaneous localization and mapping (SLAM) system that incorporates depth information and minimizes depth loss during training. Subsequently, NICER-SLAM[26] further incorporates monocular normal estimators and introduces a keyframe selection strategy. To expedite the training procedure, Nvidia proposes Instant-NGP[27], which leverages an innovative input encoding method, allowing the use of a smaller network without sacrificing quality. Despite the notable enhancement in training speed, there is no assurance of compatibility with online visual-inertial odometry (VIO) and NeRF map updates.

Among the NeRF-based localization literature, Loc-NeRF[28] introduces a real-time visual odometry (VO) algorithm by combining a particle filter with a NeRF prior map, which is trained offline. VO propagates the state of the pose, while rendered images from NeRF are used for updates. Due to the large number of particles and rendering costs from the NeRF model, Loc-NeRF operates at a much lower frequency of 0.6

Hz compared to the normal camera rate. NeRF-VINS[29] proposes a real-time VINS framework by integrating OpenVINS[8] and NeRF[24], utilizing both real and rendered images for updates at varying frequencies. *Nonetheless, none of the approaches above addresses pose initialization at the first timestamp. In other words, they assume the rigid transformation between the prior map frame and the online camera frame is known.* The only map-based relocalization work is iNeRF[30], they invert the NeRF pipeline and propose a gradient-based pose estimator by inputting a single image and a pre-trained NeRF model, but it heavily relies on a good initial guess.

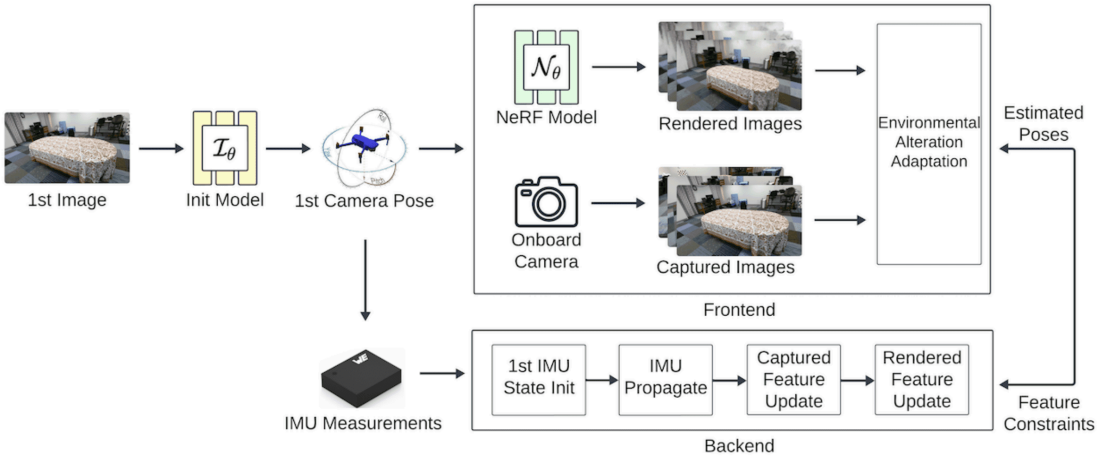


Figure 1. An overview of our NeRF-VIO framework. Commencing with the initial captured image, the pre-trained initialization model (yellow) outputs the first pose of the camera frame. Utilizing IMU integration from the timestamp of the initial IMU measurement to that of the first camera measurement, we deduce the initial IMU state backward. Throughout online traveling, we leverage both the pre-trained NeRF model (green) and the onboard camera to establish spatial constraints, facilitating the update of poses within the current sliding window. These updated poses then undergo further IMU propagation, serving as input to the NeRF model for the rendering of subsequent images.

To tackle the challenges outlined above, this paper proposes a real-time map-based VIO algorithm with pose initialization as in Fig. 1. Specifically, we introduce an initialization model to estimate the first IMU state and a NeRF model to update the poses during traveling. For the initialization, we introduce an MLP-based model, which establishes the correlations between images and poses without necessitating an initial guess. We define a novel loss function as the geodesic errors on $SE(3)$ and construct a left-invariant metric on $\mathfrak{se}(3)$. Additionally, we train a NeRF model capable of rendering images from new poses. During online traversal, the onboard camera captures images while the NeRF model renders

images based on the estimated poses from VIO. These two pipelines operate concurrently but at different frequencies. Upon receiving a new rendered image, an object removal strategy is deployed to environmental alterations between the real world and the prior map. Subsequently, both captured and rendered images are utilized to update the robot's state. The main contributions of our work include:

- We propose a novel pose estimation model to initialize the first IMU state of VINS within the prior map frame. Our approach involves training an MLP to encode the map-pose information, and defines a novel loss function as the geodesic errors on $SE(3)$. Besides, we prove the left-invariant of our proposed loss function.
- We propose an online NeRF-based VIO algorithm by integrating a NeRF-based prior map and the proposed initialization model. This algorithm utilizes both captured images from an onboard camera and rendered images from NeRF to update the state.
- We validate our proposed method using a real-world AR table dataset[31]. The results demonstrate that our initialization model surpasses *state-of-the-art* NeRF-based pose estimation solution in terms of accuracy and efficiency. Besides, our two-stage update pipeline outperforms multi-state constraint Kalman filter (MSCKF)[6] across all table sequences.

II. Preliminaries

A. NeRF Map Generation and Image Rendering

NeRF[24] employs a multilayer perceptron (MLP) to capture a radiance field representation of a scene and generate images from new perspectives. The NeRF model can be trained offline given a sequence of RGB images and the corresponding 3D location and the 2D viewing direction, where the 2D viewing direction can be expressed as a 3D Cartesian unit vector. Once we get the NeRF map $\mathcal{N}_\theta$, a new image from a novel pose can be generated and each pixel on the image is predicted by projecting a ray $\mathbf{r}$ from the center of the camera to the position of this pixel on the image plane. Then some particles are sampled uniformly within $[t_n, t_f]$ along the ray and part of them are selected based on the estimated density σ. Finally, the color value of this pixel is rendered based on those selected particles as:

$$\hat{\mathcal{C}}(\mathbf{r}) = \int_{t_n}^{t_f} \hat{T}(\mathbf{r}, t)\hat{\sigma}(\mathbf{r}, t)\hat{\mathbf{c}}(\mathbf{r}, t)dt, \tag{1}$$

where $\hat{(\cdot)}$ denotes the estimated value and $\mathbf{c}$ denotes the RGB color to be predicted at one particle. The accumulated transmittance follows $\hat{T}(\mathbf{r}, t) = \exp\Big(-\int_{t_n}^{t} \hat{\sigma}(\mathbf{r}, s)ds\Big)$. Then, the loss function can be

defined as:

$$\mathcal{L}(\mathcal{N}_\theta) = \sum_{\mathbf{r}\in\mathcal{R}} \|\hat{\mathcal{C}}(\mathbf{r}) - \mathcal{C}(\mathbf{r})\|_2^2, \tag{2}$$

where $\mathcal{R}$ denotes the set of rays. For a more comprehensive description, readers are referred to[24].

III. Problem Formulation

The goal of the NeRF-VIO is to estimate the 3D pose of the IMU frame $\{I\}$ in the global frame $\{G\}$ given an initialization model $\mathcal{I}_\theta$ and a prior map $\mathcal{N}_\theta$. Specifically, the prior map is encoded by a NeRF model, which is trained offline using the image-pose pairs from a different trajectory in the same environment. As illustrated in Fig. 2, the initialization model is designed to relocalize a captured image from a prior map, while the NeRF model renders an image based on the current pose. The initialization model runs only once before online traversal. Note that the NeRF map resides within its own world frame $\{W\}$, which is not coincident with the global frame $\{G\}$ before initialization. During online traveling, the robot updates its state using both images rendered from the NeRF map and the captured images from the onboard cameras in the camera frame $\{C\}$. We assume the sensor platform is pre-calibrated, ensuring that the relative transformation between the IMU frame and camera frame, denoted as $\mathbf{T}_I^C$, is already determined.

A. NeRF-VIO State Vector

To perform the NeRF-VIO, we include the IMU state, cloned IMU state, SLAM feature state, calibration state, and camera and IMU time-offset in the robot's state vector as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_I^\top & \mathbf{x}_{Clone}^\top & \mathbf{x}_f^\top & \mathbf{x}_{Calib}^\top & t_d \end{bmatrix}^\top, \tag{3}$$

where $t_d = t_C - t_I$ denotes the time-offset between the camera clock and the IMU clock, which treats the IMU clock as the true time. The state vector of IMU at time step k can be written as:

$$\mathbf{x}_{I_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_k}^\top & {}^G\mathbf{v}_{I_k}^\top & \mathbf{b}_{g_k}^\top & \mathbf{b}_{a_k}^\top \end{bmatrix}^\top, \tag{4}$$

where ${}^{I_k}_{G}\bar{q}$ denotes the JPL unit quaternion from the global frame to the IMU frame. ${}^G\mathbf{p}_{I_k}$ and ${}^G\mathbf{v}_{I_k}$ are the position and velocity of IMU in the global frame. $\mathbf{b}_{g_k}$ and $\mathbf{b}_{a_k}$ represent the gyroscope and accelerometer biases. During inference, the robot maintains a sliding window with m cloned IMU poses at time step k written as:

$$\mathbf{x}_{Clone_k} = \begin{bmatrix} {}^{I_{k-1}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{k-1}}^\top & \cdots & {}^{I_{k-m}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{k-m}}^\top \end{bmatrix}^\top. \tag{5}$$

In addition to the IMU state, the historical SLAM features are also stored in the state vector as:

$$\mathbf{x}_f = \begin{bmatrix} {}^{G}\mathbf{p}_{f1}^\top & \cdots & {}^{G}\mathbf{p}_{fi}^\top \end{bmatrix}^\top, \tag{6}$$

and spatial calibration between its IMU frame and camera frame will also be estimated as:

$$\mathbf{x}_{Calib_k} = \begin{bmatrix} {}^{C_k}_{I_k}\bar{q}^\top & {}^{C_k}\mathbf{p}_{I_k}^\top \end{bmatrix}^\top. \tag{7}$$

B. IMU Dynamic Model

The measurement of the IMU linear acceleration ${}^{I}\mathbf{a}_m$ and the angular velocity ${}^{I}\boldsymbol{\omega}_m$ are modeled as:

$$ {}^{I}\mathbf{a}_m = {}^{I}\mathbf{a} + {}^{I}_{G}\mathbf{R}{}^{G}\mathbf{g} + \mathbf{b}_a + \mathbf{n}_a, \tag{8}$$
$$ {}^{I}\boldsymbol{\omega}_m = {}^{I}\boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g, \tag{9}$$

where ${}^{I}\mathbf{a}$ and ${}^{I}\boldsymbol{\omega}$ are the true linear acceleration and angular velocity. $\mathbf{n}_a$ and $\mathbf{n}_g$ represent the continuous-time Gaussian noises that contaminate the IMU measurements. ${}^{G}\mathbf{g}$ denotes the gravity expressed in the global frame. Then, the dynamic system of each IMU can be modeled as in [10].

After linearization, the continuous-time IMU error-state can be written as:

$$\dot{\tilde{\mathbf{x}}}(t) \simeq \mathbf{F}(t)\tilde{\mathbf{x}}(t) + \mathbf{G}(t)\mathbf{n}(t), \tag{10}$$

where $\mathbf{F}(t)$ is the 15×15 continuous-time IMU error-state Jacobian matrix, $\mathbf{G}(t)$ is the 15×12 noise Jacobian matrix, and $\mathbf{n}(t) = [\mathbf{n}_g^\top \mathbf{n}_{wg}^\top \mathbf{n}_a^\top \mathbf{n}_{wa}^\top]^\top$ is the system noise with the covariance matrix $\mathbf{Q}$. Then, a standard EKF propagation is employed to mean and covariance[32].

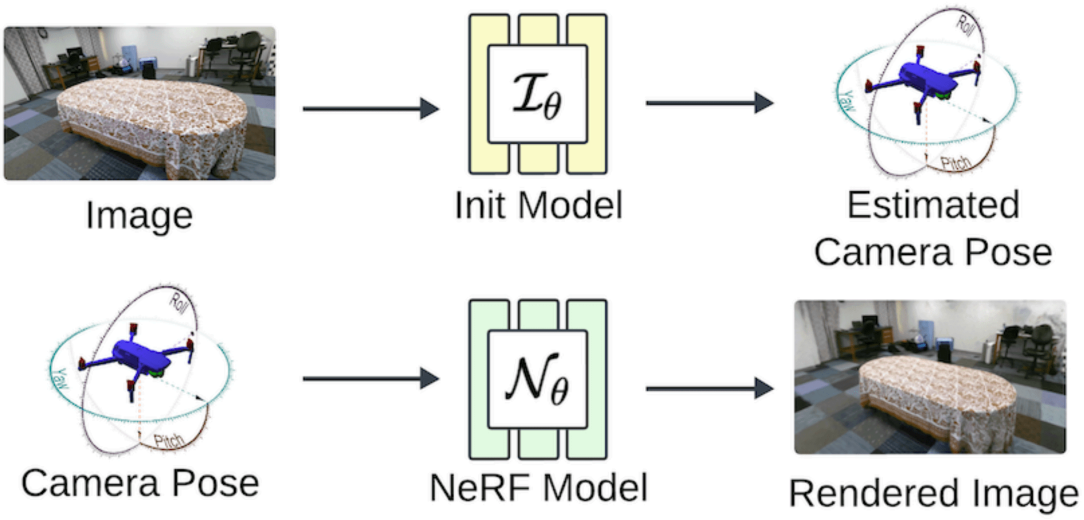


Figure 2. Comparison of input and output during model inference. The Init model estimates the camera pose in the world frame of a prior map based on a captured image. Conversely, the NeRF model renders an image when provided with a specific camera pose.

C. Initialization Model

The purpose of the initialization model is to restore the IMU state at the first timestamp $\mathbf{x}_{I_0}$ from a prior map $\mathcal{N}_\theta$. In iNeRF[30], a 6 Degrees of Freedom (DoF) pose estimation is proposed, leveraging gradient descent to reduce the residual between pixels generated from a NeRF and those within an observed image. However, this approach heavily depends on a good initial guess.

In this work, we introduce a novel MLP-based initialization model that directly maps images to poses without needing an initial estimate. Specifically, we pre-collect images and groundtruth data from the same environment and train a 7-layer MLP. This MLP encodes prior environmental knowledge, taking a sequence of images as input and outputting 6-DoF poses. Working with pose estimation in the context of $\mathfrak{se}(3)$ requires careful consideration of the underlying Lie group structure. The lack of invariance in the standard inner product on $\mathfrak{se}(3)$ has a potential drawback, as it can lead to discrepancies when comparing poses in different coordinate frames. Hence, our contribution goes beyond just initialization, as we define our loss function using geodesic distance on $SE(3)$ with a left-invariant metric. This ensures consistent and invariant pose comparisons, addressing the limitations tied to inner product-based metrics.

doi.org/10.32388/VAM5O9

In the establishment of a left-invariant metric on $SE(3)$, the definition involves specifying the inner product on the Lie algebra $\mathfrak{se}(3)$ and subsequently extending it to a Riemannian metric through left translation[33]. The left-invariant metric is established when the following equation holds[34]:

$$\langle \mathbf{x}_1, \mathbf{x}_2 \rangle_{\mathbf{S}} = \langle \mathbf{S}^{-1}\mathbf{x}_1, \mathbf{S}^{-1}\mathbf{x}_2 \rangle_{\mathbf{I}}, \tag{11}$$

where $\langle \cdot, \cdot \rangle_{\mathbf{S}}$ represents the inner product within the tangent space $T_{\mathbf{S}}SE(3)$ at an arbitrary element $\mathbf{S} \in SE(3)$, $\mathbf{x}_1, \mathbf{x}_2 \in T_{\mathbf{S}}SE(3)$, $\mathbf{I}$ denotes the identity, and $(\cdot)^{-1}$ denotes the inverse operation in the Lie group $SE(3)$.

Inspired by the definition of bi-invariant metric in $SO(3)$, the metric in $SE(3)$ can be constructed similarly. We define

$$\mathbf{M}_{\mathfrak{se}(3)} = \begin{bmatrix} \mathbf{I}_{3\times 3} & \mathbf{a} \\ \mathbf{a}^T & 1 \end{bmatrix},$$

where $\mathbf{a} \in \mathbb{R}^3$. The eigenvalues of $\mathbf{M}_{\mathfrak{se}(3)}$ are $1, 1 \pm \|\mathbf{a}\|_2$, and the condition $\|\mathbf{a}\|_2 < 1$ ensures all eigenvalues are positive. Then, the metric on $T_{\mathbf{S}}SE(3)$ is defined as:

$$\langle \mathbf{x}_1, \mathbf{x}_2 \rangle_{\mathbf{S}} = \mathrm{tr}(\mathbf{x}_1^T \mathbf{x}_2 \mathbf{M}_{\mathfrak{se}(3)}). \tag{12}$$

Lemma 1. *Left-invariant: The metric defined in (12) is left-invariant.*

Proof. For $\mathbf{S} = \begin{bmatrix} \mathbf{R}_s & \mathbf{p}_s \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3)$, let $\mathbf{x}_i \in T_{\mathbf{S}}SE(3), i = \{1, 2\}$, be $\mathbf{x}_i = \begin{bmatrix} \lfloor \omega_{i,s} \times \rfloor & \mathbf{v}_{i,s} \\ \mathbf{0} & 0 \end{bmatrix}$, we have

$$\begin{aligned} \mathbf{S}^{-1}\mathbf{x}_i &= \begin{bmatrix} \mathbf{R}_s^T & -\mathbf{R}_s^T\mathbf{p}_s \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \lfloor \omega_{i,s} \times \rfloor & \mathbf{v}_{i,s} \\ \mathbf{0} & 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_s^T \lfloor \omega_{i,s} \times \rfloor & \mathbf{R}_s^T \mathbf{v}_{i,s} \\ \mathbf{0} & 0 \end{bmatrix}. \end{aligned}$$

Then, according to (12), we have

$$\begin{aligned} &\langle \mathbf{S}^{-1}\mathbf{x}_1, \mathbf{S}^{-1}\mathbf{x}_2 \rangle_{\mathbf{I}} \\ =&\mathrm{tr}\left(\begin{bmatrix} \mathbf{R}_s^T \lfloor \omega_{1,s} \times \rfloor \mathbf{R}_s^T \mathbf{v}_{1,s} \\ \mathbf{0} \qquad\qquad 0 \end{bmatrix}^T \begin{bmatrix} \mathbf{R}_s^T \lfloor \omega_{2,s} \times \rfloor \mathbf{R}_s^T \mathbf{v}_{2,s} \\ \mathbf{0} \qquad\qquad 0 \end{bmatrix} \mathbf{M}_{\mathfrak{se}(3)} \right) \\ =&\mathrm{tr}\left(\begin{bmatrix} \lfloor \omega_{1,s} \times \rfloor^T \lfloor \omega_{2,s} \times \rfloor & \lfloor \omega_{1,s} \times \rfloor^T \mathbf{v}_{2,s} \\ \mathbf{v}_{1,s}^T \lfloor \omega_{2,s} \times \rfloor & \mathbf{v}_{1,s}^T \mathbf{v}_{2,s} \end{bmatrix} \mathbf{M}_{\mathfrak{se}(3)} \right) \\ =&\mathrm{tr}\left(\begin{bmatrix} \lfloor \omega_{1,s} \times \rfloor & \mathbf{v}_{1,s} \\ \mathbf{0} \qquad 0 \end{bmatrix}^T \begin{bmatrix} \lfloor \omega_{2,s} \times \rfloor & \mathbf{v}_{2,s} \\ \mathbf{0} \qquad 0 \end{bmatrix} \mathbf{M}_{\mathfrak{se}(3)} \right) \\ =&\langle \mathbf{x}_1, \mathbf{x}_2 \rangle_{\mathbf{S}}, \end{aligned}$$

which means that the metric is left-invariant. □

We denote f_1, f_2 as the corresponding local flows with

$$\begin{aligned} \mathbf{x}_1 &= \dot{f}_1(t) \\ \mathbf{x}_2 &= \dot{f}_2(t) \\ f_1(t) &= f_2(t) = \mathbf{S} \end{aligned} \quad . \tag{13}$$

As $f_i(t) \in SE(3)$, it can be written as:

$$f_i(t) = \begin{bmatrix} \mathbf{R}_i(t) & \mathbf{p}_i(t) \\ \mathbf{0} & 1 \end{bmatrix}, \tag{14}$$

and the corresponding twists at time t can be expressed as :

$$\mathbf{T}_i = f_i^{-1}(t)\dot{f}_i(t) = \begin{bmatrix} \lfloor \omega_i \times \rfloor & \mathbf{v}_i \\ \mathbf{0} & 0 \end{bmatrix}. \tag{15}$$

With the definition of the metric in (12), the inner product can be reformulated as:

$$\begin{aligned} \langle \mathbf{x}_1, \mathbf{x}_2 \rangle_{\mathbf{S}} &= \operatorname{tr}(\dot{f}_1^T(t)\dot{f}_2(t)\mathbf{M}_{\mathfrak{se}(3)}) \\ &= \operatorname{tr}(\mathbf{T}_1^T\mathbf{T}_2\mathbf{M}_{\mathfrak{se}(3)}) \quad \longleftarrow \text{Left Invariance} \\ &= \operatorname{tr}(\lfloor \omega_1 \times \rfloor^T \lfloor \omega_2 \times \rfloor) + \operatorname{tr}(\lfloor \omega_1 \times \rfloor^T \mathbf{v}_2 \mathbf{a}^T) \\ &\quad + \mathbf{v}_1^T \lfloor \omega_2 \times \rfloor \mathbf{a} + \mathbf{v}_1^T \mathbf{v}_2 \\ &= \begin{bmatrix} \omega_1 \\ \mathbf{v}_1 \end{bmatrix}^T \begin{bmatrix} 2\mathbf{I}_{3\times 3} & \lfloor \mathbf{a} \times \rfloor \\ \lfloor -\mathbf{a} \times \rfloor & \mathbf{I}_{3\times 3} \end{bmatrix} \begin{bmatrix} \omega_2 \\ \mathbf{v}_2 \end{bmatrix} \\ &:= \left\langle \begin{bmatrix} \omega_1 \\ \mathbf{v}_1 \end{bmatrix}, \begin{bmatrix} \omega_2 \\ \mathbf{v}_2 \end{bmatrix} \right\rangle_{\mathbf{M}_{\mathfrak{se}(3)}}. \end{aligned} \tag{16}$$

The left-invariant metric on $\mathfrak{se}(3)$ allows us to define the geodesic loss on $SE(3)$ as follows:

$$\begin{aligned} d^2(\mathbf{S}_1, \mathbf{S}_2) &= \left\langle \log_{\mathbf{S}_1}(\mathbf{S}_2), \log_{\mathbf{S}_1}(\mathbf{S}_2) \right\rangle_{\mathbf{S}_1} \\ &= \left\langle \begin{bmatrix} \lfloor \omega \times \rfloor & \mathbf{v} \\ \mathbf{0} & 0 \end{bmatrix}, \begin{bmatrix} \lfloor \omega \times \rfloor & \mathbf{v} \\ \mathbf{0} & 0 \end{bmatrix} \right\rangle_{\mathbf{S}_1} \\ &= \left\langle \begin{bmatrix} \omega \\ \mathbf{v} \end{bmatrix}, \begin{bmatrix} \omega \\ \mathbf{v} \end{bmatrix} \right\rangle_{\mathbf{M}_{\mathfrak{se}(3)}}, \end{aligned} \tag{17}$$

where $\mathbf{S}_1, \mathbf{S}_2 \in SE(3)$, $\log_{\mathbf{S}_1}(\cdot)$ represents Lie logarithm at $\mathbf{S}_1$, ω and $\mathbf{v}$ denote the rotational velocity and translational velocity from $\mathbf{S}_2$ to $\mathbf{S}_1$, respectively. Since the original data naturally lies in $\mathfrak{se}(3)$, this metric formulation offers computational advantages over the standard left-invariant metric on $SE(3)$. Specifically, it eliminates the need for mapping between $\mathfrak{se}(3)$ and $SE(3)$, while maintaining mathematical rigor through the use of the canonical inner product structure on $\mathfrak{se}(3)$. This approach both simplifies computation and preserves the geometric interpretation of the distance measure.

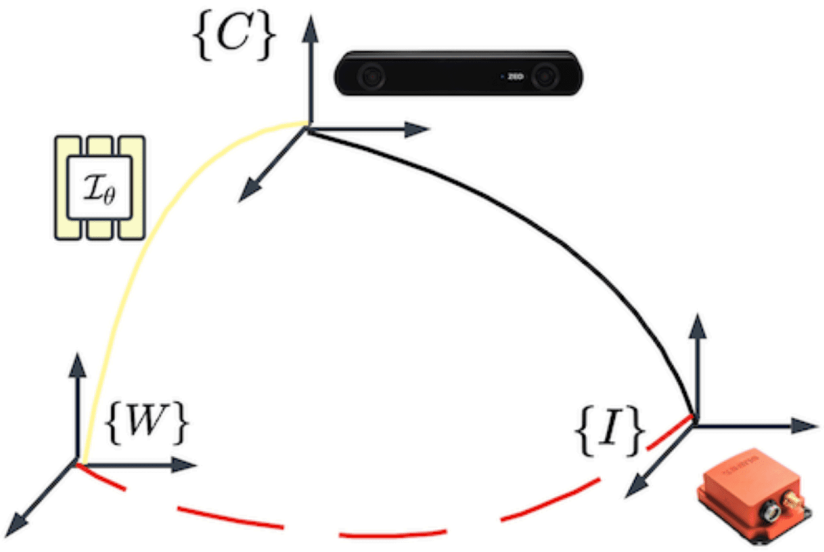


Figure 3. IMU pose initialization. From the init model, the relative pose between the first camera frame and the prior map frame can be determined. With the camera-IMU calibration parameters and the timestamps, the transformation between the first camera frame and the first IMU frame can be found.

With this loss function, we train an MLP-based initialization model to relocalize the pose of the first captured image in the prior map $\mathbf{T}_W^{C_0} := \left[{}_W^{C_0}\bar{q}^\top, {}^{C_0}\mathbf{p}_W^\top\right]$. Based on the IMU integration in (10), and the calibration parameters in (7), the relative transformation from the first IMU pose to the first camera pose $\mathbf{T}_{C_0}^{I_0} := \left[{}_{C_0}^{I_0}\bar{q}^\top, {}^{I_0}\mathbf{p}_{C_0}^\top\right]$ can be obtained. Now, the first IMU frame can be relocalized in the prior map frame as:

$$\mathbf{T}_W^{I_0} = \mathbf{T}_{C_0}^{I_0} * \mathbf{T}_W^{C_0}. \tag{18}$$

To further initialize $[{}^G\mathbf{v}_{I_0}^\top, \mathbf{b}_{g_0}^\top, \mathbf{b}_{a_0}^\top]$, we collect a window of IMU readings from timestamp 0 to the time received the first image, and initialize using the average of velocities and bias within this window.

D. Robustness to Environmental Alterations

To address dynamic objects and minor environmental alterations between the previous map and the current scenario, we introduce the grid-based Structural Similarity Index (SSIM) [35] algorithm. This method involves partitioning both the captured and rendered images into numerous small grids and computing the SSIM similarity for each grid pair across the two images as:

$$\text{SSIM}(I_x, I_y) = [l(I_x, I_y)]^{\alpha} \cdot [c(I_x, I_y)]^{\beta} \cdot [s(I_x, I_y)]^{\gamma}, \tag{19}$$

where $l(I_x, I_y)$, $c(I_x, I_y)$, and $s(I_x, I_y)$ denote the local mean (luminance), standard deviations (contrast), and cross-covariance (structural) similarity between two images. α, β, and γ denote the weights for three terms. If the similarity of a grid pair surpasses a predetermined threshold, feature extraction will be applied to both grids. Conversely, regions where the similarity falls below the threshold are deemed to contain dynamic objects, and consequently, no feature will be extracted within those areas on the rendered image.

E. Measurement Update using Captured Images

The feature measurements captured from an onboard camera can be described by:

$$\mathbf{z}_c = \Pi({}^C\mathbf{p}_f) + \mathbf{w}_c, \quad \Pi([xyz]^\top) = \begin{bmatrix} \frac{x}{z} & \frac{y}{z} \end{bmatrix}^\top, \tag{20}$$

where ${}^C\mathbf{p}_f$ denotes the position of this feature in the camera frame, and $\mathbf{w}_c$ denotes the corresponding measurement noise. Based on the estimated relative transformation between IMU and the global frame, and the estimated calibration parameters, ${}^C\mathbf{p}_f$ can be expressed as:

$${}^C\mathbf{p}_f = {}^C_I\mathbf{R}{}^I_G\mathbf{R}(\bar{t})({}^G\mathbf{p}_f - {}^G\mathbf{p}_I(\bar{t})) + {}^C\mathbf{p}_I, \tag{21}$$

where $\bar{t} = t - t_d$ is the exact camera time between the global and IMU frame.

To update a particular captured feature, we first gather all measurements of this feature within the current sliding window. Then, the measurement residuals are computed between each observation and the registered feature. By stacking all measurement residuals, we linearize them at the estimated IMU pose as follows:

$$\mathbf{r}_c = \mathbf{h}_c(\tilde{\mathbf{x}}, {}^G\tilde{\mathbf{p}}_f) + \mathbf{w}_c \simeq \mathbf{H}_{x,c}\tilde{\mathbf{x}} + \mathbf{H}_{f,c}{}^G\tilde{\mathbf{x}}_f + \mathbf{w}_c, \tag{22}$$

where $\mathbf{H}_{x,c}$ and $\mathbf{H}_{f,c}$ denote the state and measurement Jacobians of captured features, respectively. $\mathbf{w}_c$ denotes the noise vector corresponding to the captured feature. Then, the standard MSCKF update [6] is applied using left-nullspace projection for $\mathbf{H}_f$.

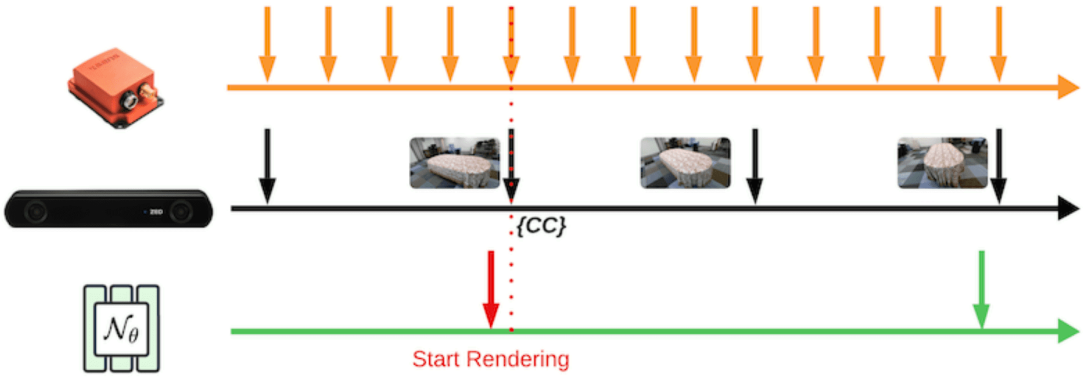


Figure 4. The three timelines denote data received from different sensors and the NeRF model. We define the closest camera frame $\{CC\}$ as the frame closest in time to when the NeRF model begins rendering.

F. Measurement Update using Rendered Images

To incorporate the observations from the rendered image and update the state vector, we aim to update the state corresponding to the pose at which the image was rendered. However, due to factors such as rendering delay and the fact that the camera pose has already been updated based on captured features, we opt for the closest camera frame $\{CC\}$ relative to the original rendered one as shown in Fig. 4. The measurement function of rendered features can be formulated as:

$$\mathbf{z}_r = \Pi \left({}^{CC}\mathbf{p}_f \right) + \mathbf{w}_r, \tag{23}$$

where $\mathbf{w}_r$ denotes the rendered noise, and

$${}^{CC}\mathbf{p}_f = {}^{CC}_{I}\mathbf{R}\,{}^{I}_{G}\mathbf{R}\left(\bar{t}\right)\left(\left({}^{G}_{W}\mathbf{R}\,{}^{W}\mathbf{p}_f + {}^{G}\mathbf{p}_W\right) - {}^{G}\mathbf{p}_I\left(\bar{t}\right)\right) + {}^{C}\mathbf{p}_I. \tag{24}$$

The error state Jacobians w.r.t. the pose of IMU can be expressed as:

$$\begin{aligned} \frac{\partial \tilde{\mathbf{z}}_r}{\partial \delta^{I}_{G}\theta} &= \mathbf{J}_\Pi\, {}^{CC}_{I}\mathbf{R}\,\lfloor {}^{I}_{G}\mathbf{R}(\bar{t})\left({}^{G}_{W}\mathbf{R}\,{}^{W}\mathbf{p}_f + {}^{G}\mathbf{p}_W - {}^{G}\mathbf{p}_I(\bar{t})\right)\times \rfloor \\ \frac{\partial \tilde{\mathbf{z}}_r}{\partial {}^{G}\tilde{\mathbf{p}}_I} &= -\mathbf{J}_\Pi\, {}^{CC}_{I}\mathbf{R}\,{}^{I}_{G}\mathbf{R}(\bar{t}), \end{aligned} \tag{25}$$

where $\mathbf{J}_\Pi$ denotes the Jacobian of perspective model.

Note that the rigid transformation $(\mathbf{C}, \mathbf{CC})$ from the initialization model is not perfect, but (24) has not modeled the initialization noise into it. Thus, we inflate the noise term as:

doi.org/10.32388/VAM5O9

$$\mathbf{w}_r' = \mathbf{w}_r + \frac{\partial \mathbf{z}_r}{\partial \delta^G_W \theta} * {}^G_W\tilde{\theta} + \frac{\partial \mathbf{z}_r}{\partial {}^G\tilde{\mathbf{p}}_W} * {}^G\tilde{\mathbf{p}}_W \tag{26}$$

where

$$\begin{aligned} \frac{\partial \mathbf{z}_r}{\partial^G_W \mathbf{R}} &= \mathbf{J}_{\Pi^{CC}_I} \mathbf{R}^I_G \mathbf{R} \left\lfloor {}^G_W\mathbf{R}\, {}^W\mathbf{p}_f \times \right\rfloor \\ \frac{\partial \mathbf{z}_r}{\partial {}^G\mathbf{p}_W} &= \mathbf{J}_{\Pi^{CC}_I} \mathbf{R}^I_G \mathbf{R}(\bar{t}) \end{aligned} \tag{27}$$

Then, the linearized model can be expressed as:

$$\mathbf{r}_r = \mathbf{h}_r \left(\tilde{\mathbf{x}}, {}^W\tilde{\mathbf{p}}_f \right) + \mathbf{w}_r' \simeq \mathbf{H}_{x,r} \tilde{\mathbf{x}} + \mathbf{H}_{f,r} {}^W\tilde{\mathbf{x}}_f + \mathbf{w}_r', \tag{28}$$

and an EKF update[6] will be employed.

	iNeRF (a)	iNeRF (b)	NeRF-VIO
Table 2	20.33 / 23.39	2.81 / 5.49	2.02 / 1.48
Table 3	9.95 / 38.37	2.70 / 4.79	2.71 / 2.04
Table 4	10.61 / 22.95	3.35 / 6.55	3.16 / 1.90
Table 5	-	5.47 / 8.09	5.21 / 4.76

Table I. The L_2 norm of the orientation / position (degrees / centimeters) of the initialization pose, utilizing iNeRF and our NeRF-VIO across AR table sequences 2-5. For iNeRF, we use different initial guesses: (**a**) a 10-degree rotational error and a 20-centimeter translation error for each axis. (**b**) a 2-degree rotational error and a 5-centimeter translation error for each axis.

	Table 2	Table 3	Table 4	Table 5
iNeRF	15.46	15.55	15.64	-
NeRF-VIO	0.11	0.12	0.13	0.11

Table II. The latency (seconds) of pose generation, utilizing iNeRF and our NeRF-VIO across AR table sequences 2-5.

IV. Experiments and Results

In this section, we validate the performance of NeRF-VIO initialization and localization using a real-world AR table dataset[31]. The dataset comprises AR table sequences 1-4, recorded around a table adorned with a textured tablecloth. Table sequence 5 introduces minor alterations by incorporating additional objects onto the table (minor environment change), while table sequences 6-8 place an additional large whiteboard to simulate the large environment change. Throughout the training process, sequence 1 is utilized to train both the initialization model and the NeRF model on an RTX 4090 GPU. In Sec. IV-A, we compare the accuracy and latency of initialization with iNeRF[30]. Rendering performance and VIO localization accuracy are evaluated and compared with MSCKF[6] in Sec. IV-B. Additionally, Sec. IV-C showcases an instance of grid-based SSIM.

(a) groundtruth (b) 1000 iterations (c) 50000 iterations (d) 200000 iterations

Figure 5. Testing results of NeRF model. (a) Groundtruth of test image. (b) Rendered image at iteration 1000. (c) Rendered image at iteration 50000. (d) Rendered image at iteration 200000.

Figure 6. Comparison of NeRF-rendered images to ground truth under normal / minor-change / large-change environments. The top row displays captured images from the closest camera frame, while the second row showcases rendered images at the same positions and orientations. Columns correspond to Table 3-6, progressing from left to right.

doi.org/10.32388/VAM5O9

	Table 2	Table 3	Table 4	Table 5	Table 6	Table 7	Average
MSCKF	1.142 / 0.034	0.750 / 0.065	2.095 / 0.077	0.656 / 0.047	0.961 / 0.049	1.161 / 0.069	1.128 / 0.057
NeRF-VIO	**0.686** / **0.023**	0.651 / 0.049	0.886 / **0.038**	**0.519** / 0.028	0.737 / 0.036	0.982 / 0.049	0.744 / 0.037
NeRF-VIO (GT Init)	0.750 / 0.024	**0.517** / **0.046**	**0.766** / 0.040	0.534 / 0.031	**0.564** / **0.028**	**0.896** / **0.043**	**0.671** / **0.035**

Table III. The ATE of the orientation / position (degrees / meters) of three VIO methods in different AR Table sequences.

A. Initialization Performance

The initialization model is trained as a 7-layer MLP using AR table sequence 1. RGB images are extracted from a Rosbag, which records from an Intel RealSense D455 camera[36]. The IMU groundtruth are captured from the Vicon system[37], and camera intrinsic and camera-IMU extrinsic parameters are calibrated using Kalibr[38]. Before forwarding the images to MLP, the corresponding camera poses are determined using 4th-order Runge-Kutta interpolation[39]. RGB images are then converted to grayscale, normalized to a range between 0 and 1, and processed through the MLP.

To compare our initialization model with iNeRF, we leverage our pre-trained NeRF model from NeRF-PyTorch[1]. as a prior map. Pose estimation of the first images in sequences 2-5 is conducted using iNeRF[2]. and our initialization model. Specifically, we initialized iNeRF with two different initial guesses: (a) a 10-degree rotational error and a 20-centimeter translation error for each axis. (b) a 2-degree rotational error and a 5-centimeter translation error for each axis. We evaluate the L_2 norm of position and orientation between estimated values and groundtruth of those two models in Table. I, while latency is provided in Table. II. We can figure out that our NeRF-VIO initialization model demonstrates superior performance over iNeRF across all four trajectories, exhibiting significantly lower latency. This can be attributed to iNeRF's reliance on gradient-based optimization, which needs to converge to local minima iteratively. Notable that we preload all models before initialization, thus the model loading time is not contained in Table. II. Additionally, iNeRF relies on a NeRF prior map, which renders it vulnerable to significant

environmental changes, leading to relocalization failures as observed in Table 5. In contrast, our model exhibits robustness to minor environmental alterations and retains the capability to reconstruct images even when a large environment changes.

B. VIO Performance

The NeRF model is constructed with 8 fully connected layers, followed by concatenation with the viewing direction of the camera, and passed through an additional fully connected layer. In addition to the image preprocessing outlined in Sec. IV-A, we further rotate the camera by 180 degrees along the x-axis to maintain consistent rendering direction. Fig. 5 illustrates the testing results of the NeRF model at various iterations during training. To evaluate the capability of NeRF-VIO to adapt to small or large environmental changes, a comparison of rendered images and ground truth is presented in Fig. 6, utilizing data from AR Table 3-6.

In assessing the impact of rendered image updates and initialization models on VIO performance, we compare our NeRF-VIO with MSCKF[6] and NeRF-VIO (GT Init), which same as NeRF-VIO but initialized from ground truth. To ensure a fair comparison, we employ the same seed and an equal number of KLT features[40] for all three methods. For NeRF-VIO, we run the NeRF rendering at 2Hz and the onboard camera at 30Hz on an Intel 9700K CPU. Table. III presents the absolute trajectory error (ATE) from Table 2-7, while Fig. 7 displays the relative pose error (RPE) of AR Table 4. It is evident that our NeRF-VIO outperforms MSCKF across all sequences and achieves performance nearly on par with the groundtruth initialization.

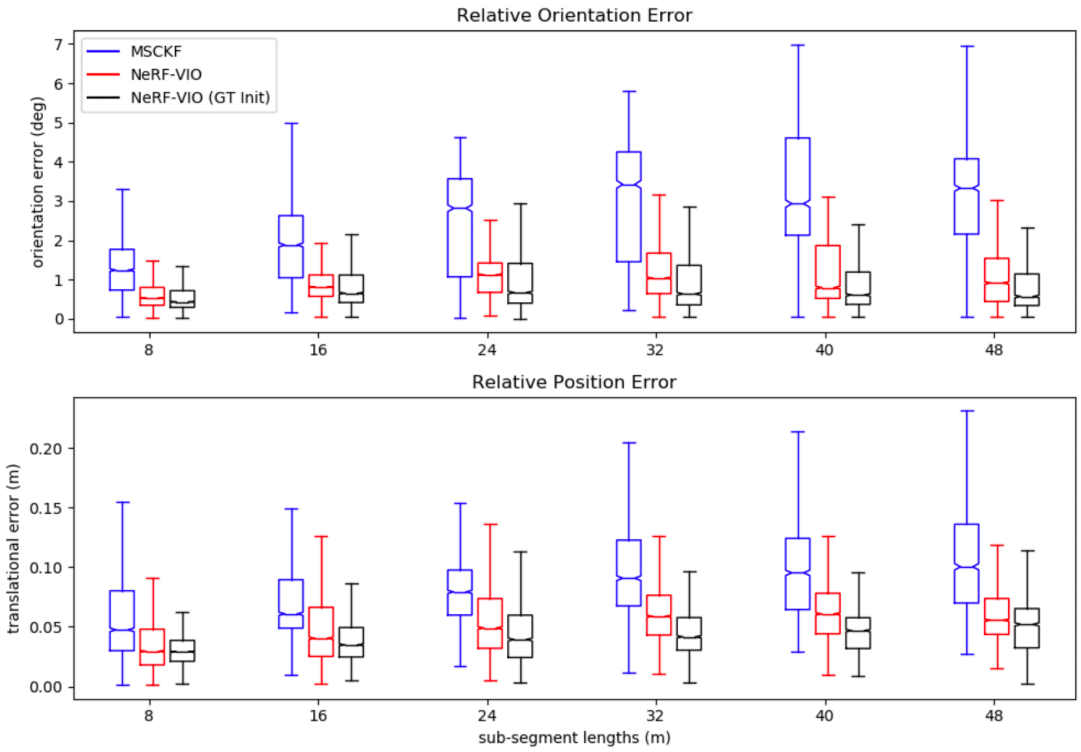
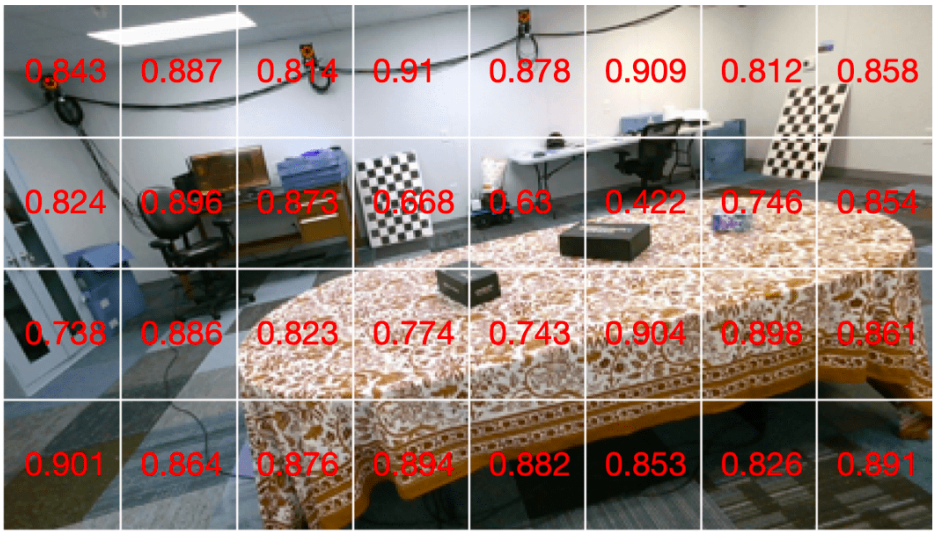


Figure 7. The RPE of MSCKF[6], NeRF-VIO (ours), and NeRF-VIO (GT Init) using AR Table 4. NeRF-VIO initializes from the pre-trained model, while NeRF-VIO (GT Init) initializes directly from groundtruth.

C. Robust to Environment Changes

To further classify the mechanics of the grid-based SSIM, we provide an example in Fig. 8 to illustrate both pixel-level and grid-level similarity. Contrasting with the third column of Fig. 6, the presence of dark pixels in Fig. 8(a) signifies a high similarity computed between the rendered and captured images. In our implementation, we assign a weight of $[1, 0.5, 0.1]$ to the exponent term in (19), and the SSIM for each grid is shown in Fig. 8(b). Then, only grids exhibiting a similarity that is larger than 0.8 are utilized for FAST[41] feature extraction. This methodology ensures consistency between the NeRF map and the real map while maintaining robustness against environmental changes.

(a) Pixel-level Similarity map



(b) Grid-level Similarity map

Figure 8. Comparison of pixel-level and grid-based SSIM. (a) A dark region denotes a high similarity, while the white region denotes a huge luminance, contrast, and structural difference weighted by $[1, 0.5, 0.1]$. (b) A grid-level similarity map is used in our algorithm. The red text denotes the similarity of each small grid.

V. Conclusions

In this paper, we have proposed a map-based visual-inertial odometry algorithm with pose initialization. We define a novel loss function for the initialization model and train an MLP model to recover the pose

doi.org/10.32388/VAM5O9

from a prior map. Besides, we proposed a two-stage update pipeline integrated into the MSCKF framework. Through the evaluation on a real-world AR dataset, we demonstrate that our NeRF-VIO outperforms all baselines in terms of both accuracy and efficiency. In the future, more attention will focus on online prior map and IMU state optimization.

Footnotes

[1] https://github.com/yenchenlin/nerf-pytorch

[2] https://github.com/salykovaa/inerf

References

1. ^*Google ARCore*.
2. ^*Apple ARKit*.
3. ^*Meta Quest 3*.
4. ^*Apple Vision Pro*.
5. ^*Wang J, Qi Y (2022). "A Multi-User Collaborative AR System for Industrial Applications". Sensors. **22** (4): 1319. [Online]. Available: https://www.mdpi.com/1424-8220/22/4/1319. PMID 35214221*.
6. a, b, c, d, e, f, g*Mourikis AI, Roumeliotis SI (2007). "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation." In: Proceedings 2007 IEEE International Conference on Robotics and Automation. pp. 3565-3572. doi:10.1109/ROBOT.2007.364024*.
7. ^*Qin T, Li P, Shen S (2018). "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator". IEEE Transactions on Robotics. **34** (4): 1004–1020. doi:10.1109/TRO.2018.2853729*.
8. a, b*Geneva P, Eckenhoff K, Lee W, Yang Y, Huang G (2020). "OpenVINS: A Research Platform for Visual-Inertial Estimation." In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 4666-4672. doi:10.1109/ICRA40945.2020.9196524*.
9. ^*Zhu P, Yang Y, Ren W, Huang G (2021). "Cooperative Visual-Inertial Odometry." In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 13135-13141. doi:10.1109/ICRA48506.2021.9561674*.
10. a, b*Zhang Y, Zhu P, Ren W (2023). "PL-CVIO: Point-Line Cooperative Visual-Inertial Odometry." In: 2023 IEEE Conference on Control Technology and Applications (CCTA). pp. 859-865. doi:10.1109/CCTA54093.2023.10253266*.

11. ^a, ^b*Galvez-López D, Tardos JD (2012). "Bags of Binary Words for Fast Place Recognition in Image Sequence s." IEEE Transactions on Robotics. **28** (5): 1188–1197. doi:10.1109/TRO.2012.2197158.*

12. ^*Siam SM, Zhang H (2017). "Fast-SeqSLAM: A fast appearance based place recognition algorithm." In: 2017 IEEE International Conference on Robotics and Automation (ICRA). pp. 5702-5708. doi:10.1109/ICRA.2017.7989671.*

13. ^*Chen Z, Jacobson A, Sünderhauf N, Upcroft B, Liu L, Shen C, Reid I, Milford M. "Deep learning features at scale for visual place recognition." In: 2017 IEEE International Conference on Robotics and Automation (ICRA); 2017. p. 3223-3230. doi:10.1109/ICRA.2017.7989366.*

14. ^a, ^b*Mur-Artal R, Montiel JMM, Tardf3s JD (2015). "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". IEEE Transactions on Robotics. **31** (5): 1147–1163. doi:10.1109/TRO.2015.2463671.*

15. ^a, ^b*Mur-Artal R, Tardós JD (2017). "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras". IEEE Transactions on Robotics. **33** (5): 1255–1262. doi:10.1109/TRO.2017.2705103.*

16. ^*Kasyanov A, Engelmann F, Stfcckler J, Leibe B (2017). "Keyframe-based visual-inertial online SLAM with relocalization." In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 6662-6669. doi:10.1109/IROS.2017.8206581.*

17. ^*Schneider T, Dymczyk M, Fehr M, Egger K, Lynen S, Gilitschenski I, Siegwart R (2018). "Maplab: An Open Framework for Research in Visual-Inertial Mapping and Localization." IEEE Robotics and Automation Letters. **3** (3): 1418-1425. doi:10.1109/LRA.2018.2800113.*

18. ^*Sarlin P-E, Cadena C, Siegwart R, Dymczyk M (2019). "From Coarse to Fine: Robust Hierarchical Localization at Large Scale." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.*

19. ^*Geneva P, Huang G (2022). "Map-based Visual-Inertial Localization: A Numerical Study." In: 2022 International Conference on Robotics and Automation (ICRA). pp. 7973-7979. doi:10.1109/ICRA46639.2022.9811829.*

20. ^*Cramariuc A, Bernreiter L, Tschopp F, Fehr M, Reijgwart V, Nieto J, Siegwart R, Cadena C (2023). "maplab 2.0 – A Modular and Multi-Modal Mapping Framework". IEEE Robotics and Automation Letters. **8** (2): 520–527. doi:10.1109/LRA.2022.3227865.*

21. ^*Zhang Z, Jiao Y, Huang S, Xiong R, Wang Y (2023). "Map-Based Visual-Inertial Localization: Consistency and Complexity." IEEE Robotics and Automation Letters. **8** (3): 1407–1414. doi:10.1109/LRA.2023.3239314.*

22. ^*Liu L, Li H, Dai Y (2017). "Efficient Global 2D-3D Matching for Camera Localization in a Large-Scale 3D Map." In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), Oct 2017.*

23. ^ Geppert M, Liu P, Cui Z, Pollefeys M, Sattler T (2019). "Efficient 2D-3D Matching for Multi-Camera Visual Localization." In: 2019 International Conference on Robotics and Automation (ICRA). pp. 5972-5978. doi:10.1109/ICRA.2019.8794280.

24. a, b, c, d Mildenhall B, Srinivasan PP, Tancik M, Barron JT, Ramamoorthi R, Ng R (2020). "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis." In: ECCV.

25. ^ Zhu Z, Peng S, Larsson V, Xu W, Bao H, Cui Z, Oswald MR, Pollefeys M. "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2022 June. p. 12786-12796.

26. ^ Zhu Z, Peng S, Larsson V, Cui Z, Oswald MR, Geiger A, Pollefeys M. "NICER-SLAM: Neural Implicit Scene Encoding for RGB SLAM." In: 2024 International Conference on 3D Vision (3DV); 2024. p. 42-52. doi:10.1109/3DV62453.2024.00096.

27. ^ Müller T, Evans A, Schied C, Keller A (2022). "Instant neural graphics primitives with a multiresolution hash encoding". ACM Transactions on Graphics (ToG). **41** (4): 1–15.

28. ^ Maggio D, Abate M, Shi J, Mario C, Carlone L. "Loc-NeRF: Monte Carlo Localization using Neural Radiance Fields." In: 2023 IEEE International Conference on Robotics and Automation (ICRA); 2023. p. 4018-4025. doi:10.1109/ICRA48891.2023.10160782.

29. ^ Katragadda S, Lee W, Peng Y, Geneva P, Chen C, Guo C, Li M, Huang G (2024). "NeRF-VINS: A Real-time Neural Radiance Field Map-based Visual-Inertial Navigation System." In: 2024 IEEE International Conference on Robotics and Automation (ICRA). pp. 10230-10237. doi:10.1109/ICRA57147.2024.10610051.

30. a, b, c Yen-Chen L, Florence P, Barron JT, Rodriguez A, Isola P, Lin T-Y (2021). "iNeRF: Inverting Neural Radiance Fields for Pose Estimation." In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1323-1330. doi:10.1109/IROS51168.2021.9636708.

31. a, b Chen C, Geneva P, Peng Y, Lee W, Huang G (2023). "Monocular Visual-Inertial Odometry with Planar Regularities." In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 6224-6231. doi:10.1109/ICRA48891.2023.10160620.

32. ^ Trawny N, Roumeliotis SI (2005). "Indirect Kalman filter for 3D attitude estimation". University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep. **2**: 2005. Citeseer.

33. ^ ***Petersen P***. Riemannian geometry. Springer; 2006. Vol. 171.

34. ^ Gallier J. Geometric methods and applications: for computer science and engineering. Springer Science & Business Media; 2011. Vol. 38.

35. ^Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004). "Image quality assessment: from error visibility to structural similarity". IEEE Transactions on Image Processing. **13** (4): 600-612. doi:10.1109/TIP.2003.819861.

36. ^Intel RealSense D455. Available from: https://www.intelrealsense.com.

37. ^Vicon.

38. ^Rehder J, Nikolic J, Schneider T, Hinzmann T, Siegwart R (2016). "Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes." In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 4304-4311. doi:10.1109/ICRA.2016.7487628.

39. ^Butcher JC (1996). "A history of Runge-Kutta methods". Applied Numerical Mathematics. **20** (3): 247–260. doi:10.1016/0168-9274(95)00108-5.

40. ^Lucas BD, Kanade T (1981). "An Iterative Image Registration Technique with an Application to Stereo Vision." In: IJCAI'81: 7th international joint conference on Artificial intelligence, vol. 2, Vancouver, Canada, Aug. 1981, pp. 674-679. PDF. HAL hal-03697340.

41. ^Viswanathan DG. "Features from accelerated segment test (fast)". In: Proceedings of the 10th workshop on image analysis for multimedia interactive services, London, UK; 2009. p. 6-8.

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.