NeRF-VINS: A Real-time Neural Radiance Field Map-based Visual-Inertial Navigation System

Saimouli Katragadda - saimouli@udel.edu
Woosik Lee - woosik@udel.edu
Yuxiang Peng - yxpeng@udel.edu
Patrick Geneva - pgeneva@udel.edu
Chuchu Chen - ccchu@udel.edu
Guoquan Huang - ghuang@udel.edu

Department of Computer Science
University of Delaware, Delaware, USA

RPNG

Robot Perception and Navigation Group (RPNG)
Supplement - RPNG-2023-NERF
Last Updated - Sept 18, 2023

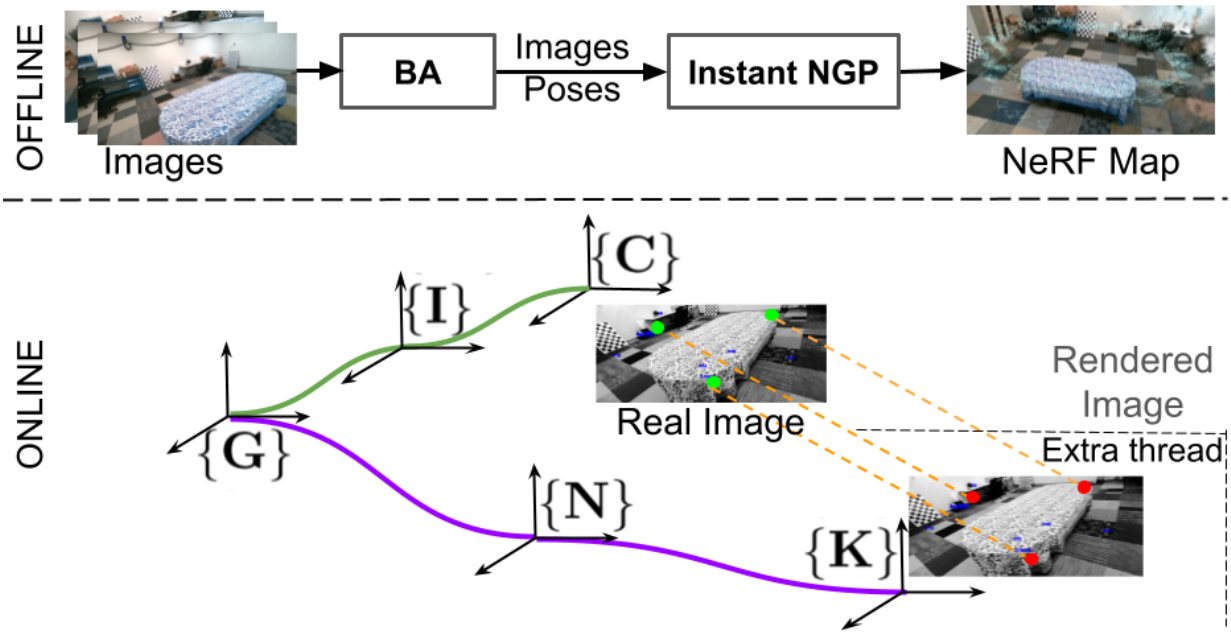

Figure 1: Overview of the proposed NeRF-VINS, where $\{G\}$ is the global VIO frame, $\{N\}$ is the map frame, $\{K\}$ denotes the NeRF rendered image. $\{I\}$ and $\{C\}$ are IMU and camera frame, respectively. For full algorithm refer to Alg. 1

This document supplements our paper, cited in [1], by offering further insights and results regarding our proposed system.

In section 1, we describe our estimator choice followed by how we fuse NeRF Map measurements in detail along with system algorithm 1.

1 NeRF-Visual Inertial Navigation System Estimator Choice

In the realm of visual-inertial localization, two primary paradigms emerge, each with distinct trade-offs in terms of accuracy and computational efficiency. One approach adopts a graph optimization-based methodology, where visual and inertial measurements are formalized as a graph, and iterative optimization techniques are employed to estimate the camera pose, landmarks (features), and IMU states. This paradigm, exemplified by works such as [2], excels in accuracy, as it directly fuses visual and inertial information in an iterative manner. However, it tends to be computationally demanding due to the iterative optimization process. Conversely, another approach is filter-based, integrating camera and IMU measurements within a filter framework. This approach is known for its computational efficiency while maintaining a reasonable level of accuracy. Notably, the multi-state constraint Kalman filter (MSCKF) and its variants, as presented in [3], [4], [5], are popular choices within this paradigm. Unlike some traditional filter-based methods, MSCKF efficiently handles the integration of camera and IMU data while retaining the ability to provide accurate estimates. It's important to clarify that while MSCKF is a form of an EKF, its distinction lies in its efficient handling of features without explicitly incorporating them into the state vector. This feature-efficiency trade-off sets it apart from traditional EKF SLAM approaches that can experience cubic growth in numerical complexity as the number of features increases. This key advantage positions MSCKF as an efficient choice within the filter-based paradigm for visual-inertial localization.

It's important to note that both paradigms have their strengths and weaknesses, and the choice between them should consider the specific requirements of the application, weighing the trade-offs between accuracy and computational efficiency. While the graph optimization-based approach offers higher accuracy, it often requires more computational resources due to its iterative nature. In contrast, the filter-based approach, especially exemplified by MSCKF, balances accuracy and efficiency, making it a compelling choice for resource-constrained real-time visual-inertial localization.

Given the choice of the estimator, we now focus on the formulation of the estimator which is built on top of OpenVINS [6].

At time t_k, the system state $\mathbf{x}_k$ consists of the current inertial navigation states $\mathbf{x}_{I_k}$, historical IMU poses (clones) $\mathbf{x}_{T_k}$, and a subset of 3D environmental point features, $\mathbf{x}_f$:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{I_k}^\top & \mathbf{x}_{T_k}^\top & \mathbf{x}_f^\top \end{bmatrix}^\top \tag{1}$$

$$\mathbf{x}_{I_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_k}^\top & {}^G\mathbf{v}_{I_k}^\top & \mathbf{b}_g^\top & \mathbf{b}_a^\top \end{bmatrix}^\top \tag{2}$$

$$\mathbf{x}_{T_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_k}^\top & \cdots & {}^{I_{k-c}}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_{k-c}}^\top \end{bmatrix}^\top \tag{3}$$

$$\mathbf{x}_f = \begin{bmatrix} {}^G\mathbf{p}_{f_1}^\top & \cdots & {}^G\mathbf{p}_{f_i}^\top \end{bmatrix}^\top \tag{4}$$

where ${}^I_G\bar{q}$ is the unit quaternion (${}^I_G\mathbf{R}$ in rotation matrix form) that represents the rotation from the global $\{G\}$ to the IMU frame $\{I\}$; ${}^G\mathbf{p}_I$, ${}^G\mathbf{v}_I$, and ${}^G\mathbf{p}_{f_i}$ are the IMU position, velocity, and i'th point feature position in $\{G\}$; $\mathbf{b}_g$ and $\mathbf{b}_a$ are the gyroscope and accelerometer biases.

$$\mathbf{x}_{I_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_k}^\top & {}^G\mathbf{v}_{I_k}^\top & \mathbf{b}_g^\top & \mathbf{b}_a^\top \end{bmatrix}^\top \tag{5}$$

$$\tag{6}$$

We choose to store IMU clones instead of camera clones because IMU data offers advantages like high frequency, low latency, and robustness in challenging scenarios, which are crucial for accurate and responsive localization.

1.1 IMU Measurement Model

IMUs measure accelerations and angular rates at a higher frequency (400 Hz) than cameras (30 Hz). By integrating IMU data, we estimate the device's position and velocity changes between camera frames. This prediction step aids tracking and pose estimation, especially in scenarios with limited or no visual feature information, making IMU an ideal choice for propagation. The IMU kinematics are used to evolve the state from time t_k to t_{k+1}:

$${}^I_G\dot{\bar{q}}(t) = \frac{1}{2}\mathbf{\Omega}(\boldsymbol{\omega}(t)){}^I_G\bar{q}(t) \tag{7}$$

$${}^G\dot{\mathbf{p}}_I(t) = {}^G\mathbf{v}_I(t) \tag{8}$$

$${}^G\dot{\mathbf{v}}_I(t) = {}^{I(t)}_{G}\mathbf{R}^\top\mathbf{a}(t) \tag{9}$$

$$\dot{\mathbf{b}}_g(t) = \mathbf{n}_{wg}(t) \tag{10}$$

$$\dot{\mathbf{b}}_a(t) = \mathbf{n}_{wa}(t) \tag{11}$$

where $\boldsymbol{\omega}(t) = [\omega_1\ \omega_2\ \omega_3]^\top$ and $\mathbf{a}(t)$ are the angular velocity and acceleration in the IMU local frame $\{I\}$; $\mathbf{\Omega}(\boldsymbol{\omega}(t)) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^\top & 0 \end{bmatrix}$ where $\lfloor\cdot\rfloor$ is the skew-symmetric matrix. $\mathbf{n}_{wg}$ and $\mathbf{n}_{wa}$ are white

Algorithm 1 `NeRF Map-based VINS`

Propagation: Section 1.1

- Propagate the IMU navigation state estimate $\hat{\mathbf{x}}_{k+1|k}$ based on (15), and the state covariance based on (16).

Feature Tracking: For an incoming image

- Perform stochastic cloning of the current state [7]
- Extract visual key points and descriptors from the current real image and match to previous real image
- Render NeRF image at the current camera pose + 10cm offset in x-axis and extract keypoints and match with current real image (see Fig. 2)

Update: Section 1.2.3

- Perform MSCKF update for VIO and NeRf features (i.e those that have lost their tracks)
- Initialize new SLAM features if needed and perform EKF update

Gaussian noise that drive the IMU biases. A canonical three-axis IMU provides linear acceleration and angular velocity measurements, ${}^{I}\mathbf{a}_m$ and ${}^{I}\boldsymbol{\omega}_m$, expressed in the local IMU frame $\{I\}$ modeled as:

$$\mathbf{a}_m(t) = \mathbf{a}(t) + {}^{I}_{G}\mathbf{R}(t){}^{G}\mathbf{g} + \mathbf{b}_a(t) + \mathbf{n}_a(t) \tag{12}$$

$$\boldsymbol{\omega}_m(t) = \boldsymbol{\omega}(t) + \mathbf{b}_g(t) + \mathbf{n}_g(t) \tag{13}$$

where ${}^{G}\mathbf{g} \simeq [0, 0, -9.8]^\top$ is the gravitational acceleration expressed in $\{G\}$, $\mathbf{n}_g$ and $\mathbf{n}_a$ are zero-mean white Gaussian noise. ${}^{I}_{G}\mathbf{R}$ denotes the rotation matrix from global frame to local IMU frame. The IMU nonlinear kinematics can be formulated as a function of:

$$\mathbf{x}_{I_{k+1}} = \mathbf{f}_I\left(\mathbf{x}_{I_k}, {}^{I}\mathbf{a}_k, {}^{I}\boldsymbol{\omega}_k, \mathbf{n}_I\right) \tag{14}$$

where $\mathbf{n}_I = [\mathbf{n}_g^\top \ \mathbf{n}_a^\top \ \mathbf{n}_{wg}^\top \ \mathbf{n}_{wa}^\top]^\top$. Linearize the IMU kinematics Eq. (14) we can get:

$$\tilde{\mathbf{x}}_{k+1} = \boldsymbol{\Phi}_k \tilde{\mathbf{x}}_k + \mathbf{G}_k \mathbf{n}_I \tag{15}$$

where $\boldsymbol{\Phi}_k$ is the linearized state translation matrix and $\mathbf{G}_k$ is the noise Jacobian.

The computed $\boldsymbol{\Phi}_k$ and $\mathbf{G}_k$ is used to propagate the covariance from t_k to t_{k+1}

$$\mathbf{P}_{k+1|k} = \boldsymbol{\Phi}(\mathbf{t_{k+1}}, \mathbf{t_k})\mathbf{P_{k|k}}\boldsymbol{\Phi}(\mathbf{t_{k+1}}, \mathbf{t_k})^\top + \mathbf{G_k}\mathbf{Q_d}\mathbf{G_k}^\top \tag{16}$$

1.2 Measurement Model

Our camera observes environmental features as it moves along its trajectory. For a feature we consider the following relation to our state:

1.2.1 Camera Measurement Model

$$\mathbf{z}_{C_k} = \begin{bmatrix} u_n \\ v_n \end{bmatrix} + \mathbf{n}_{C_k} \tag{17}$$

$$= \mathbf{h}(\mathbf{x}_{T_k}, {}^{G}P_f) + \mathbf{n}_{C_k} \tag{18}$$

$$= \boldsymbol{\Lambda}({}^{C_k}\mathbf{p}_f) + \mathbf{n}_{C_k} \tag{19}$$

$${}^{C_k}\mathbf{p}_f = {}^{C}_{I}\mathbf{R}^{I_k}_{G}\mathbf{R}({}^{G}\mathbf{p}_f - {}^{G}\mathbf{p}_{I_k}) + {}^{C}\mathbf{p}_I \tag{20}$$

where $\mathbf{\Lambda}([x\ y\ z]^\top) = [x/z\ y/z]^\top$, and $\mathbf{z}_{n,k}$ is the normalized feature bearing. $\mathbf{n}_{C_k}$ is the white Gaussian noise. Here we can assume we know the camera distortion parameters to recover the normalized pixel coordinates, $\mathbf{z}_{n,k}$, and that the extrinsic transform between the camera and IMU, $\{{}^C_I\mathbf{R}, {}^C\mathbf{p}_I\}$, is known with reasonable accuracy.

Linearizing Eq. (17) gives the following measurement residual:

$$\mathbf{r}_{C_k} = \mathbf{z}_{C_k} - \mathbf{h}_c(\hat{\mathbf{x}}_{T_k}, {}^G\hat{\mathbf{p}}_f) \simeq \mathbf{H}_T\tilde{\mathbf{x}}_{T_k} + \mathbf{H}_f{}^G\tilde{\mathbf{p}}_f + \mathbf{n}_{C_k} \tag{21}$$

where $\mathbf{H}_T$ and $\mathbf{H}_f$ are the Jacobian matrix of the measurement with respect to each state.

1.2.2 NeRF Measurement Model

When a camera image reading is received, a NeRF render is triggered at a pose with a small horizontal positional offset (e.g., 10 cm, as in our experiments) from the current camera pose. This strategy draws inspiration from human perception, emulating the principles of stereo vision where two eyes yield subtly distinct viewpoints, thereby enhancing triangulation accuracy. Furthermore, this method capitalizes on the considerable image overlap, promoting resilient feature matching even when camera is static—an essential component for precise location estimation. Once the rendering is completed, descriptor-based feature matching is performed to the current image, where a 2D-to-2D prior keyframe measurement model is leveraged [8]. For example, consider that from the rendered image we get a bearing measurement, $\mathbf{z}_{N_k}$, which is related to the state as:

$$\mathbf{z}_{N_k} = \mathbf{h}_n\left({}^Gn\mathbf{p}_f\right) + \mathbf{n}_{N_k} =: \mathbf{\Lambda}({}^K\mathbf{p}_f) + \mathbf{n}_{N_k} \tag{22}$$

$${}^K\mathbf{p}_f = {}^N\mathbf{p}_f + s{}^N_K\mathbf{R}({}^N\mathbf{p}_G + {}^N_G\mathbf{R}{}^G\mathbf{p}_f) \tag{23}$$

where s is the scale factor of the map and $\mathbf{n}_{N_k}$ is the zero mean Gaussian noise. Note that we model the bearing as only a function of the feature ${}^G\mathbf{p}_f$, and consider the map transform $\{s, {}^G_N\mathbf{R}, {}^Np_G\}$ to be known and the rendered camera pose $\{{}^N_K\mathbf{R}, {}^K\mathbf{p}_N\}$ to have some known orientation and position error $\{{}^G_N\tilde{\theta}, {}^N\tilde{p}_G\}$

Thus, we have the following linearized model:

$$\mathbf{r}_{N_k} = \mathbf{z}_{N_k} - \mathbf{h}_n({}^G\hat{\mathbf{p}}_f) = s\mathbf{H}_{\mathbf{\Lambda}}{}^K_N\mathbf{R}{}^N_G\mathbf{R}{}^G\tilde{\mathbf{p}}_f + \mathbf{n}'_{N_k} \tag{24}$$

where $\lfloor\cdot\times\rfloor$ is the skew-symmetric matrix and

$$\mathbf{n}'_{N_k} = s\mathbf{H}_{\mathbf{\Lambda}}{}^K_N\mathbf{R}(\lfloor{}^N_G\mathbf{R}{}^G\mathbf{p}_f\times\rfloor{}^N_G\tilde{\boldsymbol{\theta}} + {}^N\tilde{\mathbf{p}}_G) + \mathbf{n}_{N_k} \tag{25}$$

1.2.3 EKF Update

For features that are tracked longer than the current sliding window, initialized into the active state as SLAM features and perform EKF udpate. Note that SLAM features will not remain active forever, instead they will be marginalized out for computational savings. On the other hand, for features that are short lived are considered as MSCKF features and updated accordingly [3]. The measurement linearized model can be used to update the features in the state or can be stacked with the real image measurements (26) to perform (SLAM or MSCKF) EKF update.

$$\begin{bmatrix}\tilde{\mathbf{z}}_{Ck}\\ \tilde{\mathbf{z}}_{Nk}\end{bmatrix} = \begin{bmatrix}\mathbf{H}^i\\ \mathbf{0}\end{bmatrix}\tilde{\mathbf{x}}_C + \begin{bmatrix}\mathbf{H}^C_f\\ \mathbf{H}^N_f\end{bmatrix}{}^G\tilde{\mathbf{p}}_f + \begin{bmatrix}\mathbf{n}_{Ck}\\ \mathbf{n}'_{N_k}\end{bmatrix} \tag{26}$$

$$\tag{27}$$

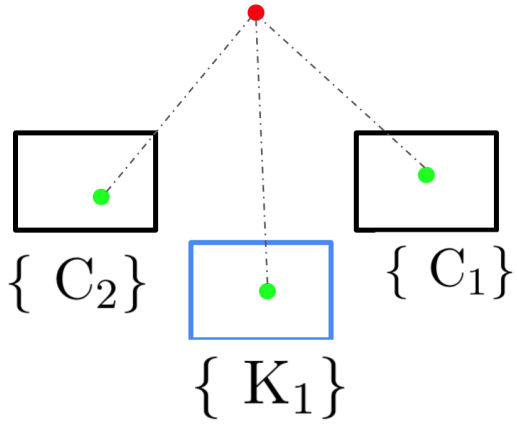

Figure 2: Assuming NeRF Map image $\{\mathbf{K_1}\}$ features matches to actively tracked feature of cloned frames $\{\mathbf{C_1}, \mathbf{C_2}\}$, additional NeRF feature measurements are added to the feature tracks to form an implicit loop closure constraint.

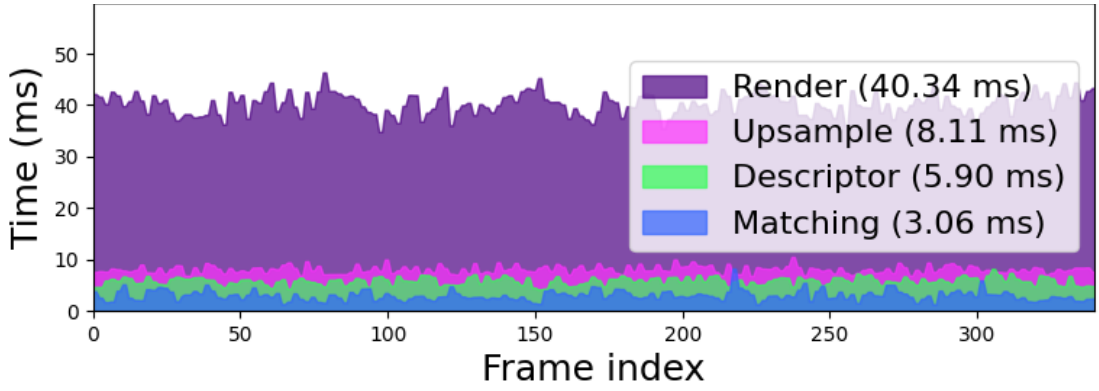

Figure 3: Pipeline computation time running on Jetson Orin

1.3 Image Rendering and Optimization

Rendering NeRF images remains a computationally expensive operation, especially on embedded devices like the Jetson AGX Orin. It takes approximately 660 ms (2Hz) to render an image with dimensions 424 × 240. To improve render speed and minimize loop-closure latency, we use a two-step process. First, we generate NeRF to render at half the original resolution, resulting in images sized at 212 × 140 pixels. While this reduces computational overhead, it also introduces a lower image resolution. The significance of this lower resolution becomes evident when we consider feature matching between the NeRF-rendered images and real images. Feature matching relies on the precise alignment of distinctive points, or keypoints, in two images. Lowering the resolution can cause a misalignment of these keypoints, leading to inaccurate matching and, subsequently, compromising the performance of our visual-inertial navigation system. To address this, we employ an up-sampling technique using FSRCNN [9] to restore the NeRF renders to their original size. This up-sampling process not only enhances the visual quality of the images but also aligns them with the higher-resolution real images, ensuring that feature matching is performed accurately. By striking this balance between computational speed and image quality, not only accelerates the rendering process but also preserves the fidelity necessary for feature matching. We further enhance performance by reducing resolution levels and the hashing size of the model in InstantNGP

(a) DBoW (b) NetVLAD (c) Ours

Figure 4: Qualitative study of failure cases of classical place recognition method. Green and Red lines indicate inliers and outliers, respectively. Input image (left of each column) and retrieved at full resolution , rendered for the NeRF case. These are full resolution images rendered offline.

[10]. Additionally, we minimize multiple copy times to the CPU by directly transferring images to our frontend superpoint-based descriptors for fast extraction. The rendering is run on a separate thread to prevent blocking of the real-time VINS. The SuperPoint feature matching network has been modified to use a lightweight ResNet18 [11] and optimized to support a 16-bit floating point using the TensorRT pipeline [12] to further improve performance. This secondary thread which performs rendering and matching runs at an adequate speed of 10 to 15Hz on the Jetson to aid in real-time estimation (Fig 3).

2 Additional Results

Unlike the results presented in the paper, the rendered images depicted in Fig. 4 are at full resolution (848×480). If rendering speed is not a constraint, we tend to observe a higher percentage of inliers. This observation is logical because NeRF consistently delivers better matches, and higher resolution facilitates higher quality feature detections, resulting in a higher percentage of inliers. Exploring this avenue further, particularly in experimenting with Gaussian splatting [13], holds promise, given its inherently fast rendering capabilities.

Table 1: HLoc and DBoW failure rates

	table_01	**table_02**	**table_03**	**table_04**	**table_05**	**table_06**	**table_07**
table_01 only	0/2506	1039/2914	142/7006	12/6068	1777/6164	1251/2767	1182/4784
table_01+table_05	0/2506	1039/2914	142/7006	12/6068	0/6164	0/2767	0/4784
DBoW+2Dto2D	6/2205	1542/2613	1650/6705	142/5768	4722/5863	1977/2466	3340/4484

TODO: include oV 2 clones, 3 clones setup

2.1 Failure Rates of Different Localization Pipelines

We present failure rates of HLoc and DBoW in Tab. 1. The NeRF map is trained on a subset of the table01 dataset and was used to test table 2, 3, 4 and Table5 has different environment. Therefore, the map was created using table 5 and used to test table 5, 6, and 7 labeled as "table01+table05".

"table01 only" indicates table1 map is used to test the rest of the tables. Errors larger than 5 degrees and 0.1 meters are considered as failure case for HLoc (*degree/meter*), for reference, failure cases in the current DBoW Pipeline DBoW+2Dto2D are also recorded. The failure definition in the current pipeline is when query images cannot find 30 matches with the top 5 retrieved images, it is considered as a failure case. The total is different since the current pipeline skips the first several seconds for proper initialization of our system.

2.2 Minimum Test Setup

The more features (in our case slam features) and clones (historical IMU states) we use the more the state size grows and can become computationally expensive to perform an update. To test the limits of our system we used minimum clones to push the limits of our pipeline with just 2 clones with no slam features. See Tab. 2 for results, even with just 2 clones and no slam features our pipeline was stable enough to provide drift free localization. The higher the clones the more stable the system when there are no feature matches and hence contributing to better overall accuracy of our system. We also included 12 clones setup and different slam configuration of our baseline system (i.e openvins) and optimization based method VINS-fusion with and without loop closures for comparison. We noticed that for map based localization descriptor based matching works the best instead of the optical flow like KLT (nerf feat0 slam 5clones klt). Approaches that optimize the photometric loss such as KLT tend to not perform well if images have large baseline in between them and tend to be sensitive to large motions and viewpoints. For this reason we opted for descriptor based matching.

Table 2: ATE Table (degree / m) noise $n_k = 1$ was used

	table_01	**table_02**	**table_03**	**table_04**	**table_05**	**table_06**	**table_07**	**Average**
nerf_feat_0slam_2clones	0.425 / 0.012	0.470 / 0.008	0.362 / 0.009	0.415 / 0.020	0.765 / 0.108	0.574 / 0.041	0.707 / 0.015	0.553 / 0.039
nerf_feat_0slam_5clones	0.443 / 0.013	0.437 / 0.010	0.357 / 0.009	0.366 / 0.030	0.439 / 0.028	0.543 / 0.038	0.739 / 0.015	0.475 / 0.022
nerf_feat_0slam_5clones_klt	0.550 / 0.021	1.528 / 0.051	0.416 / 0.024	0.438 / 0.027	0.464 / 0.026	0.608 / 0.031	0.824 / 0.019	0.650 / 0.028
nerf_feat_0slam_8clones	0.450 / 0.013	0.444 / 0.008	0.365 / 0.009	0.364 / 0.017	0.433 / 0.027	0.602 / 0.040	0.743 / 0.015	0.470 / 0.019
openvins_0slam	0.832 / 0.057	1.058 / 0.029	1.291 / 0.063	0.672 / 0.066	1.078 / 0.067	0.725 / 0.047	1.712 / 0.079	1.053 / 0.059
openvins_25slam	0.458 / 0.036	0.964 / 0.033	1.068 / 0.035	0.822 / 0.038	1.394 / 0.057	0.821 / 0.038	1.321 / 0.055	0.978 / 0.042
openvins_50slam	0.692 / 0.044	0.546 / 0.025	1.079 / 0.058	1.192 / 0.035	0.523 / 0.026	0.988 / 0.039	0.889 / 0.041	0.844 / 0.038
mono_vinsfusion_vio	1.619 / 0.058	1.319 / 0.030	1.468 / 0.076	1.748 / 0.056	1.122 / 0.034	0.978 / 0.053	1.670 / 0.093	1.870 / 0.079
mono_vinsfusion_loop	0.662 / 0.035	0.980 / 0.066	0.611 / 0.046	0.764 / 0.029	0.903 / 0.019	0.592 / 0.019	1.534 / 0.039	0.864 / 0.036

Table 3: RPE Table

	8m	**16m**	**24m**	**32m**	**40m**	**48m**
nerf_feat_0slam_2clones	0.482 / 0.034	0.477 / 0.031	0.521 / 0.039	0.475 / 0.035	0.512 / 0.039	0.482 / 0.038
nerf_feat_0slam_5clones	0.479 / 0.029	0.469 / 0.024	0.513 / 0.031	0.464 / 0.024	0.500 / 0.028	0.481 / 0.024
nerf_feat_0slam_5clones_klt	0.612 / 0.041	0.638 / 0.035	0.668 / 0.043	0.561 / 0.035	0.558 / 0.039	0.510 / 0.030
nerf_feat_0slam_8clones	0.477 / 0.028	0.472 / 0.023	0.505 / 0.030	0.459 / 0.023	0.487 / 0.027	0.475 / 0.022
openvins_0slam	0.786 / 0.046	1.113 / 0.062	1.522 / 0.077	1.738 / 0.097	1.874 / 0.110	1.858 / 0.125
openvins_25slam	0.698 / 0.039	0.976 / 0.046	1.278 / 0.054	1.543 / 0.060	1.717 / 0.065	1.899 / 0.074
openvins_50slam	0.673 / 0.038	0.859 / 0.044	1.060 / 0.049	1.247 / 0.050	1.424 / 0.059	1.515 / 0.067
mono_vinsfusion_vio	0.863 / 0.050	1.461 / 0.075	2.192 / 0.110	3.075 / 0.145	4.197 / 0.181	5.586 / 0.208
mono_vinsfusion_loop	0.553 / 0.044	0.615 / 0.032	0.699 / 0.051	0.703 / 0.033	0.749 / 0.052	0.847 / 0.044

References

[1] 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". In: (2023). arXiv: 2309.09295 [cs.RO].

[2] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. "Keyframe-based visual–inertial odometry using nonlinear optimization". In: *The International Journal of Robotics Research* (2015).

[3] Anastasios I. Mourikis and Stergios I. Roumeliotis. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007.

[4] Mingyang Li and Anastasios I. Mourikis. "High-precision, consistent EKF-based visual-inertial odometry". In: *The International Journal of Robotics Research* (2013).

[5] Patrick Geneva, James Maley, and Guoquan Huang. "An Efficient Schmidt-EKF for 3D Visual-Inertial SLAM". In: *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, June 2019.

[6] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. "OpenVINS: A Research Platform for Visual-Inertial Estimation". In: *Proc. of the IEEE International Conference on Robotics and Automation*. Paris, France, 2020. URL: https://github.com/rpng/open_vins.

[7] S.I. Roumeliotis and J.W. Burdick. "Stochastic cloning: a generalized framework for processing relative state measurements". In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 2. 2002, 1788–1795 vol.2. DOI: 10.1109/ROBOT.2002.1014801.

[8] Patrick Geneva, Kevin Eckenhoff, and Guoquan Huang. "A Linear-Complexity EKF for Visual-Inertial Navigation with Loop Closures". In: *Proc. International Conference on Robotics and Automation*. Montreal, Canada, May 2019.

[9] Chao Dong, Chen Change Loy, and Xiaoou Tang. "Accelerating the super-resolution convolutional neural network". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer. 2016.

[10] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". In: *ACM Trans. Graph.* (2022).

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[12] NVIDIA. *TensorRT*. https://github.com/NVIDIA/TensorRT.

[13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. "3D Gaussian Splatting for Real-Time Radiance Field Rendering". In: *ACM Transactions on Graphics* (2023).