2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) | 978-1-6654-0158-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISMAR52148.2021.00043

RNIN-VIO: Robust Neural Inertial Navigation Aided Visual-Inertial Odometry in Challenging Scenes

Danpeng Chen[1,2], Nan Wang[2], Runsen Xu[1], Weijian Xie[1,2], Hujun Bao[1], and Guofeng Zhang[1*]
[1]State Key Lab of CAD&CG, Zhejiang University
{runsenxu, baohujun, zhangguofeng}@zju.edu.cn
[2]SenseTime Research and Tetras.AI
{chendanpeng, wangnan, xieweijian}@tetras.ai

ABSTRACT

In this work, we propose a tightly-coupled EKF framework for visual-inertial odometry with NIN (Neural Inertial Navigation) aided. Traditional VIO systems are fragile in challenging scenes with weak or confusing visual information, such as weak/repeated texture, dynamic environment, fast camera motion with serious motion blur, etc. It is extremely difficult for a vision-based algorithm to handle these problems. So we firstly design a robust deep learning based inertial network (called RNIN), using only IMU measurements as input. RNIN is significantly more robust in challenging scenes than traditional VIO systems. In order to take full advantage of vision-based algorithms in AR/VR areas, we further develop a multi-sensor fusion system RNIN-VIO, which tightly couples the visual, IMU and NIN measurements. Our system performs robustly in extremely challenging conditions, with high precision both in trajectories and AR effects. The experimental results of evaluation on dataset evaluation and online AR demo demonstrate the superiority of the proposed system in robustness and accuracy.

Index Terms: VIO—SLAM—INS—IMU; AR—6DoF—Motion Tracking—;

1 INTRODUCTION

6DoF motion tracking is a key technology for many applications, such as robot navigation, pedestrian navigation, AR/VR, etc. In the area of AR, high-precision and robust camera position tracking are required to improve the immersive experience, and the system must run in real-time with low-power consumption. Visual inertial navigation [2, 11, 18, 18, 19, 23, 27, 35] can achieve high precision and robustness on a mobile platform using a consumer-grade camera and a low-cost IMU. At present, a series of excellent AR developer platforms in the industry have adopted similar visual-inertial tracking solutions, such as Apple's ARKit[1], Google's ARCore[2], which have been widely used and achieved amazing results.

Although the existing visual-inertial navigation has achieved state-of-the-art accuracy and stability, there are still many difficult problems in the application of consumer-level AR. The current visual-inertial navigation scheme always relies on continuous and reliable visual tracking, which easily fails to track when the visual information is quite weak or confusing, such as the scenes with weak/repeated textures, shimmering light or dynamic change, the camera is moving very fast with serious motion blur, and so on. The IMU is not affected by external visual information and forms a complement to the visual sensor. The purpose of our work is to make better use of IMU information to reduce the dependence of vision information in VIO system and provide higher accuracy and more robust motion tracking.

*Corresponding author: Guofeng Zhang

[1]https://developer.apple.com/augmented-reality/arkit/
[2]https://developers.google.com/ar

The IMU measures the acceleration and angular velocity of the device's movement and the acceleration data must be integrated twice to get the position information. However, consumer-grade IMU data contains various noises such as bias, which will bring a significant error accumulation. Traditional methods use zero-velocity update (ZUPT) [13] and zero angular rate update [28] or step counting [3, 16] to reduce the cumulative error. But these methods only work well under periodic pedestrian motion. With the development of deep learning, data-driven methods have been used in inertial navigation. Deep learning based methods use windowed IMU measurements to robustly estimate relative displacement or velocity. IONet [6] is the first to propose an LSTM based network to learn patterned motion and estimate the relative displacement of the window. RoNIN [14] assumes the rotation can be read from the phones and uses a network to regress the velocity. TLIO [21] uses EKF to fuse the IMU raw data and the relative displacement measurement from the ResNet network. The experiments have shown that the neural network based methods have more advantages than traditional methods both in accuracy and robustness.

In summary, compared with vision based methods, neural network inertial navigation does not rely on visual information and can provide more robust observations. Inspired by these methods, we try to integrate the robust but low-precision neural network IMU observations with the traditional visual-inertial navigation system, in order to construct a highly robust and precise visual-inertial navigation system. The traditional visual-inertial fusion can estimate high accuracy states, such as pose, velocity, the direction of gravity, and IMU bias. On the one hand, better state estimation can provide better initialization data for the IMU neural network to regress relative translation and covariance, thereby improving the effectiveness of network estimation. On the other hand, the IMU neural network can enhance the robustness of the navigation system. In summary, our paper has the following three major contributions:

- We propose a novel deep learning based inertial network to learn the regularity of humans' motion patterns in time series. The designed relative loss and absolute loss can make the network care about the local accuracy as well as the long-term global accuracy. Our experiments show the proposed neural inertial network has state-of-the-art precision.
- As far as we know, we are the first to propose a tightly-coupled multi-sensor fusion motion tracking system to fuse vision, IMU and network measurements. Leveraging high-robust inertial networks, our visual-inertial fusion system can dramatically improve the robustness in extreme situations and also remain high precision in normal scenes at the same time.
- All of our algorithms can run on a consumer-grade mobile phone in real-time. The mobile AR application demonstrates the advantages of the proposed system.

We evaluate our system both on open source data IDOL [33] and our own collected data. The experimental results show that our deep learning based network of IMU data has achieved state-of-the-art accuracy. Moreover, we integrate the deep learning based network

978-1-6654-0158-6/21/$31.00 ©2021 IEEE
DOI 10.1109/ISMAR52148.2021.00043

with vision and inertial data to improve the accuracy and robustness of visual-inertial navigation. Finally, we run the proposed system on a consumer-grade mobile device to demonstrate the effectiveness and efficiency. Compared with ARCore, our method achieves better accuracy and robustness, especially under some extremely challenging conditions.

2 Related Work

2.1 Visual Related Odometry

We briefly outline the works strictly focused on monocular camera and inertial measurement unit data.

Visual odometry: Visual odometry uses image information to incrementally estimate ego-motion of a camera. PTAM [17] is the first to propose a VSLAM system based on keyframe optimization. The whole system is divided into a camera tracking thread and a global mapping thread. In the tracking thread, 3D-2D matching is used to obtain the matching relationship between the current image and the local map, and then the optimal pose of the current camera is obtained by minimizing the reprojection error. The global mapping thread uses bundle adjustment to eliminate the error of keyframes and map points. For the consideration of speed and robustness, SVO [12] directly uses image intensity for matching, instead of slow feature extraction process. It also uses deep probability filtering to improve the accuracy of map point estimation. RKSLAM [20] is another keyframe based VSLAM. By using the prior of homograph and planes, the robustness of tracking has a significant improvement. With the high processing frame rate, it is very suitable for mobile AR. DSO [8] uses the direct method to take full use of image information. It tries to minimize photography error to track camera pose. This method is robust in the condition of motion blur but heavily depends on image quality on the other hand. ORB-SLAM [24] is a state-of-the-art visual SLAM system, which divides the system into three modules: camera tracking, local mapping, and loop closure. At the same time, it proposes a robust visual initialization method. Although VSLAM achieves relatively high accuracy in an ideal visual environment, it often fails under dynamic conditions, severe lighting changes, fast movement, pure rotation motion, etc. Monocular VSLAM cannot estimate the absolute scale of the scene and will lead to scale drift.

Visual inertial odometry: To alleviate visual dependence and estimate absolute scale information, researchers have proposed a visual-inertial navigation system, which combines the motion information of vision and IMU to obtain high-precision and robust positioning results. In the past few decades, visual inertial odometry (VIO) can be divided into filtering based methods and optimization based methods. A representative of the filtering methods is MSCKF [19, 23]. MSCKF uses multi-state Kalman filtering to fuse visual geometric information and IMU motion information. This method does not include the visual landmark in the state vector and has low computational complexity. It obtains high accuracy in a variety of scenarios. SR-ISWF [35] is another typical filtering method. Because of the square-root formulation, they can use single-precision floating points to speed up numerical operations while ensuring numerical properties. The typical representative of the optimization method is OKVIS. OKVIS [18] is a tightly-coupled, nonlinear optimization-based method, which minimizes visual reprojection errors and preintegrated IMU errors. It uses sliding window marginalization strategies to bound the computational cost. VINS-Mono [27] is a robust visual-inertial SLAM system. It has robust initialization and 4DoF global pose graph optimization methods to improve accuracy. Although the VIO system achieves impressive accuracy and robustness, it needs continuous and effective visual information. It will fail in the case of long-range perspective, dramatic lighting changes, weak textures, and continuous visual loss, etc.

2.2 IMU based Odometry

There are many IMU based motion tracking algorithms in literature, which are named PDR (Pedestrian Dead Reckoning) or INS (Inertial navigation system). We divide these algorithms into two categories:

Traditional kinematics based algorithms: These methods include acceleration integration approach [15] and step counting approach [3]. The former one first subtracts the gravity component, gets the user's acceleration, and integrate them twice to get user translation. But this approach is difficult to be applied in mobile phones, because consumer-grade IMU is noisy and large error will accumulate easily. Since pedestrians have a regular walking pattern, the latter one [3, 16] detect steps with acceleration directly. And step lengths are estimated by heuristic rules. Then it counts steps to get the translation. This method is used widely on mobile platforms since it does not depend on very accurate acceleration measurement.

Data-driven algorithms: With the rapid development of deep learning, data-driven methods are also applied in the area of INS. Deep learning based method can robustly estimate velocity or relative translation using IMU measurements directly rather than using double integration, which will lead to dramatical accumulated error on consumer-grade phones. [4–7, 9, 10, 14, 21, 25, 32, 34] are typical examples of data driven methods. IONet [6] first proposed an LSTM network structure to estimate relative displacement and accumulate them to obtain the 2D position. Silva do Monte Lima et al. [32] process Gyroscope and accelerometer data separately by two convolutional layers. The output of these layers is concatenated and fed to LSTM layers to regressed relative pose. However, a simple two-layer convolution cannot extract good high-level features of motion, and the loss function of a single-window cannot enable the network to learn the motion relationship of a long sequence. RoNIN [14] assumes orientation is known from android phones and IMU data are transformed into a gravity-aligned coordinate firstly. Then it uses three variants of neural network based on ResNet, LSTM, or TCN to regresses velocity and integrates them together. TLIO [21] adopts traditional EKF to fuse raw IMU measurements and displacement which are learned from ResNet neural network. This approach reduces yaw and position drift observably and can output 6DoF pose at IMU frame rate. Dugne-Hennequin et al. [7] presented a deep analysis of data-driven inertial odometry with DNN that could help the community in future work. Our network of deep inertial odometry is inspired by TLIO and RoNIN. Current human motion hidden variables are learned by the ResNet module. However, We believe human motion is continuous and regular. So we use LSTM to fuse the current hidden state with the previous hidden state and produce the best current hidden state of motion, which will be passed through two fully connected layers to regress the relative position and the corresponding covariance. The loss functions are newly designed to make the network care about the local accuracy and the long-term global accuracy.

The above method is mainly applied to human movement, and there are also methods to apply the learned inertial odometer to the scene of the vehicles. AbolDeepIO [9] uses three LSTMs to extract features from accelerometer, gyroscope, and sampling time. It concatenates features as input of LSTM to regress the amplitude of the 3D relative displacement and rotation. But it cannot predict 3D position from these outputs. AI-IMU [4] uses a CNN network to dynamically estimate the uncertainty of the simple model assumptions of vehicle motion and combines EKF to estimate the vehicle's position, speed, and IMU bias. Obviously, this method is only suitable for automotive application scenarios. Inspired by the loop control system, a feedback control system is proposed [25], which takes the output of the neural network at the previous moment and the acceleration of the car as input to predict the current displacement of the car. And experiments show that the system has great advantages compared with traditional INS.

Recently, there are other works using deep inertial networks to

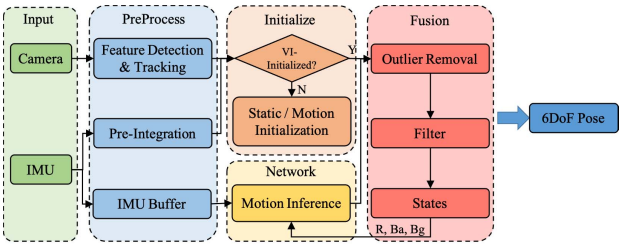


Fig. 1: The pipeline of the proposed system.

estimate orientation. OriNet [10] uses a neural architecture based on LSTMs to estimate the rotation and a Genetic Algorithm algorithm is introduced to correct the gyroscope bias. Weber et al. [34] designed a neural network based on RNN or CNN to directly regress attitude from IMU data. Martin Brossard et al. [5] use CNN to correct the noise and bias of the gyroscope, and directly integrate the corrected gyroscope to calculate the attitude. Experiments of [5,10,34] demonstrate that the proposed network is better than traditional filtering algorithms under different motions. Because the traditional IMU orientation estimation algorithm can achieve relatively good results in a variety of sports, we do not consider the problem of attitude estimation but focus on the more difficult translation estimation problem from IMU measurements.

3 Overview

3.1 System overview

The structure of the proposed system is illustrated in Fig. 1. Our estimation system takes IMU data and image as input and is mainly composed of four modules including preprocessing, initialization, neural network, and filter.

As in [27], the IMU data between two consecutive frames are pre-integrated and sparse features are extracted and tracked from the images. We also maintain the IMU Buffer as the input of the neural network.

An initialization phase is necessary to ensure the filter will converge. In order to adapt to a variety of motion situations, when the system detects static motion, we use static initialization, and when the system detects motion, we use motion initialization.

The neural network is trained to learn prior motion distribution. The network takes a local window of IMU data as input, without obtaining the initial velocity, and regresses the 3D relative displacement and uncertainty of this window. Regardless of the influence of noise, under the same windowed IMU data, different initial velocities correspond to different motions, which means that motion cannot be estimated by IMU data alone. Since our system is mainly designed for handheld AR, AR glasses, and other applications, our estimated movement is mainly concentrated on human motions. We believe that despite the broad movement distribution, the human movement distribution should be relatively narrow, and the same IMU data corresponding to different motions will rarely appear. Based on this consideration, we believe that such a network can work normally, just like the previous related work.

The filter propagates with IMU data and uses sparse features and network outputs for updates, which tightly couples all measurements. In our system, the visual constraints can be removed at any time, and state estimation can also be carried out only based on IMU measurements.

3.2 Notation

We now define notations and frame definitions that we use throughout the paper. We consider ${}^{W}(\cdot)$ as the world frame. ${}^{C}(\cdot)$ is the camera frame and ${}^{I}(\cdot)$ is the body frame or IMU frame.We use both rotation matrices R and Hamilton quaternions q to represent rotation. $\otimes$ represents the multiplication operation between two quaternions. ${}^{W}\boldsymbol{g} = [0,0,g]^{T}$ is the gravity vector in the world frame. The direction of the gravity is aligned with the z axis of world frame. Finally, we denote $\hat{(\cdot)}$ the noisy measurement or estimate of a certain quantity.

4 Learning Motion Model

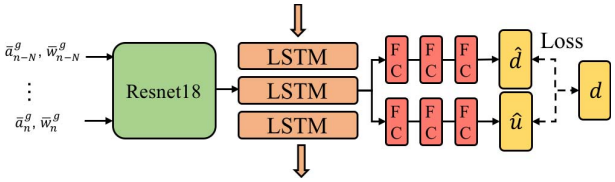


Fig. 2: The proposed nerual inertial network.

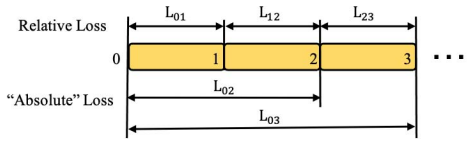


Fig. 3: The proposed loss function.

4.1 Network Architecture

The overall architecture of our network is shown in Fig. 2, which consists of the 1D version of ResNet18, standard LSTM, and fully connected layers. Similar to TLIO [21], we use the ResNet module to learn human motion hidden variables. However, we believe that human motion is continuous and regular. So we use LSTM to fuse the current hidden state with the previous hidden state to estimate the best current hidden state of motion. Finally, two fully connected layers are used to regress the relative position of the window and the corresponding covariance.

The mathematical form of the network is:

$$
\begin{aligned}
(\hat{\boldsymbol{d}}, \hat{\boldsymbol{u}}) &= f(({}^{W}\boldsymbol{a}_{n-N}, {}^{W}\boldsymbol{w}_{n-N}), ..., ({}^{W}\boldsymbol{a}_{n}, {}^{W}\boldsymbol{w}_{n}), \boldsymbol{h}_{n-N}) \\
{}^{W}\boldsymbol{a}_{n} &= {}^{W}R_{n}(\boldsymbol{a} - \boldsymbol{b}_{a}) - {}^{W}\boldsymbol{g} \\
{}^{W}\boldsymbol{w}_{n} &= {}^{W}R_{n}(\boldsymbol{w} - \boldsymbol{b}_{g})
\end{aligned}
\tag{1}
$$

where $f(\cdot)$ is the function defined by the neural network. $\boldsymbol{a}$ and $\boldsymbol{w}$ are the raw acceleration and angular velocity read from the IMU sensor, and $\boldsymbol{b}_a$, $\boldsymbol{b}_g$ are the corresponding bias, which are obtained from the following filter section 5. ${}^{W}\boldsymbol{g}$ is the gravity vector $[0,0,9.8]$. ${}^{W}\boldsymbol{a}_{n-N}, {}^{W}\boldsymbol{w}_{n-N}$ respectively represent the acceleration and angular velocity of human motion at the Nth IMU moment in the inertial frame. $\boldsymbol{h}_{n-N}$ is a hidden state produced by the LSTM at the last time step. At each time step, the network infers motion based on a hidden state $\boldsymbol{h}_{n-N}$, a local window of N acceleration and angular velocity in the inertial frame, which are similar to the acceleration and angular speed of human motion with noise. The output of the network contains two vectors: the relative displacement $\hat{\boldsymbol{d}}$ and their uncertainties $\hat{\boldsymbol{u}}$.

4.2 Loss Function

To make the network care about the local accuracy and also pay attention to the long-term global accuracy, We design two different loss functions shown in Fig.3.

Relative Loss In order to allow the network to learn the movement of a single window and improve the measurement accuracy of

the direct output of the network, we add a relative loss (RL) function to the single window. The Mean Square Error (MSE) is chosen to optimize the loss. RL loss is defined as:

$$L_{RL}^{MSE}(\boldsymbol{d},\hat{\boldsymbol{d}}) = \frac{1}{n}\sum_{i=n}^{n}(\sum_{j=m}^{m+M}(\boldsymbol{d}_{j\to j+1} - \hat{\boldsymbol{d}}_{j\to j+1})) \tag{2}$$

where $\hat{\boldsymbol{d}}_{j\to j+1}$ is the j-th window 3D translation output of the network and $\boldsymbol{d}_{j\to j+1}$ is the corresponding ground truth. n is batch size during training, and m is the start window of sequence. M is the number of LSTM window.

'Absolute' Loss In addition to the accuracy of a single window, we are more concerned about the cumulative error over a longer period of time. To allow the network to learn the long sequence relationship of human motion, thereby reducing the long-term cumulative error, we designed the absolute loss function

$$L_{AL}^{MSE}(\boldsymbol{d},\hat{\boldsymbol{d}}) = \frac{1}{n}\sum_{i=n}^{n}(\sum_{j=m}^{m+L}(\boldsymbol{d}_{m\to j+1} - \hat{\boldsymbol{d}}_{m\to j+1})) \tag{3}$$

To obtain the mean and covariance of Gaussian, fused into the EKF framework, we also use a negative log-likelihood loss to replace the above MSE loss during training the neural network.

$$\boldsymbol{L}^{NLL} = \frac{1}{2}(\boldsymbol{d}-\hat{\boldsymbol{d}})^T\hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{d}-\hat{\boldsymbol{d}}) + \frac{1}{2}ln(\left|\hat{\boldsymbol{\Sigma}}\right|) \tag{4}$$

Our covariance estimation is similar to [21, 30]. The output of the covariance of the network is a three-dimensional vector. The three elements are the log of diagonal elements of the covariance matrix.

4.3 Data Collection

Our dataset is mainly composed of two parts, including IDOL open source 20 hours of data and 7 hours of data collected by ourselves. IDOL data were collected by Kaarta Stencil, which uses a lidar-visual-inertial SLAM algorithm to estimate its pose at 100Hz. From testing in a VICON motion capture studio, they measured $< 1.5°$ RMS orientation error and $< 10cm$ RMS position error. IDOL mainly includes some simple plane movements. To increase the diversity of sports and equipment, we use multiple smartphones, such as Huawei, Xiaomi, OPPO, etc., to collect data with cameras and IMUs. The full dataset was captured by five people and includes a variety of sports, including walking, running, standing still, going up and down stairs, and random shaking, etc. We use BVIO introduced in Section 5 to provide the positions aligned with gravity at IMU frequency on the dataset. Part of the data is collected in the VICON room, so it has high-precision trajectory provided by VICON. The dataset is divided into 80% train, 10% validation, 10% test subsets. Our self-collected data is available at the project page: https://zju3dv.github.io/rnin-vio/.

4.4 Implementation Details

For the input of the network, we use a sliding window with a certain sampling frequency on each data sequence to collect input samples. Each sliding window includes N IMU samples and the total input dimension is $N \times 6$. In our training, the sampling frequency is selected as 20Hz. On each sequence, we perform linear interpolation at 100 Hz to ensure that all input data are at a fixed frequency. Our purpose is to allow the network to learn the distribution of human movement, therefore all IMUs are processed according to Equations (1). The window size is 100, which represents 1.0s, and the LSTM time steps is selected as 10, so the size of each input sample is $10 \times 100 \times 6$. The supervision signal of the network is the relative position of the window under the gravity system.

Data enhancement is a commonly used method to increase the diversity of data and avoid over-fitting to a certain extent. Because different IMU devices may have different degrees of Gaussian white noise and deviation, and to widely adapt to the noise of a variety of IMU equipment, in the training phase, we randomly superimpose Gaussian white noise and bias on each input sample. We randomly sample 1×6 acceleration bias and gyroscope bias from a uniform distribution, and two random Gaussian noises with dimensions of $10 \times 100 \times 3$ for each input sample $10 \times 100 \times 6$. There is also a potential error in the direction of gravity, so we perform gravity noise enhancement like the method [21]. Since people have the same motion distribution in different yaw directions, we add a random yaw angle rotation to each sample so that the network can learn the yaw angle invariance characteristics.

We use Pytorch [26] as the implementation of our model, and train it through the Adam optimizer with an initial learning rate of 0.0001. We found that if the NLL loss is directly used for training, the network will be difficult to converge and the accuracy will be poor. So we first use MSE loss to train until convergence, and then switch to NLL loss training until convergence. On the IDOL data set, a total of about 150 epochs are required to fully converge. It takes about 10 hours of training time on an NVIDIA 1080Ti GPU. The model with the best validation loss was chosen as the best model for testing.

5 Multi-Sensor Fusion for Robust 6DoF Tracking

Our multi-sensor fusion motion tracking system tightly integrates vision, IMU and neural inertial network measurements. The objective of the system is to minimize the cost function $C_{k+1}^{\oplus}$ for all the measurements at each time step $k+1$:

$$C_{k+1}^{\oplus} = C_k + C_u + C_z + C_n \tag{5}$$

where C_k, C_u, C_z, C_n represents the cost function of the prior, IMU measurements, visual measurements and neural inertial network information. And $\{C_k, C_u, C_z\}$ is the same with the cost terms of traditional MSCKF based VIO. We use square root inverse filter [22, 35] to solve the cost function $C_{k+1}^{\oplus}$ because of its low cost in computation. Similar to [35], BVIO (short for Basic VIO) maintain a sliding window poses in filter state and employ square root inverse filter to fuse cost functions of $\{C_k, C_u, C_z\}$. The major difference with [35] is that BVIO does not consider SLAM features. By adding neural inertial network constraints to the BVIO system, we achieve the complete system RNIN-VIO. We will first show the state vector of the system, and then describe step by step how to obtain $C_{k+1}^{\oplus}$. Finally, we will show how the system updates the state frame by frame.

5.1 State Definition

Like most EKF based VIO systems, we define the states of our system as follows:

$$\boldsymbol{S}_k = [\boldsymbol{S}_{I_{k-(m-1)}}^T, ..., \boldsymbol{S}_{I_k}^T, \boldsymbol{S}_{E_k}^T]^T \tag{6}$$

for $i = k-(m-1), ..., k$ represent the state vector of m cloned IMU poses at frame i. Each cloned IMU pose is defined as:

$$\boldsymbol{S}_{I_i} = [{}^W\boldsymbol{q}_{I_i}^T, {}^W\boldsymbol{p}_{I_i}^T]^T \tag{7}$$

where ${}^W\boldsymbol{q}_{I_i}^T$ is the orientation of frame i with the quaternion representation in the world frame, ${}^W\boldsymbol{p}_i$ is the position of frame i in world frame. And the last term $\boldsymbol{S}_{E_k}^T$ is defined as follows:

$$\boldsymbol{S}_{E_k} = [\boldsymbol{b}_{g_k}^T, \boldsymbol{b}_{a_k}^T, {}^W\boldsymbol{v}_{I_k}^T]^T \tag{8}$$

where $\boldsymbol{b}_{g_k}^T$ and $\boldsymbol{b}_{a_k}^T$ correspond to the bias of gyroscope and accelerater. ${}^W\boldsymbol{v}_k^T$ is the velocity of frame k in the world coordinate.

The error state is defined as $\tilde{\boldsymbol{S}}_k = \boldsymbol{S}_k - \hat{\boldsymbol{S}}_k$, where $\boldsymbol{S}_k$ is the true state and $\hat{\boldsymbol{S}}_k$ is the estimated state.

5.2 IMU Measurement

IMU noise model: IMU (Inertial Measurement Unit) is consisted of accelerometer and gyroscope, which can measure the device's rotational velocity and linear acceleration at a high frame rate. But the measurements are always with white Gaussian noise and time-varying biases in consumer-level mobile phones. The biases of both are modelled as random walks, driven by zero-mean Gaussian noise processes $\boldsymbol{n}_{g_k}$ and $\boldsymbol{n}_{a_k}$, while the gyroscope and accelerometer measurements $\hat{\boldsymbol{w}}_k$ and $\hat{\boldsymbol{a}}_k$ at frame k are:

$$\begin{aligned}\hat{\boldsymbol{w}}_k &= \boldsymbol{w}_k + \boldsymbol{b}_{g_k} + \boldsymbol{n}_{g_k} \\ \hat{\boldsymbol{a}}_k &= \boldsymbol{a}_k + \boldsymbol{b}_{a_k} + {}^{I}\boldsymbol{R}_{W}\,{}^{W}\boldsymbol{g} + \boldsymbol{n}_{a_k}\end{aligned} \tag{9}$$

where $\boldsymbol{w}_k$ and $\boldsymbol{a}_k$ are ground truth states at frame k.

Pre-integration: IMU measurements have a high frame rate (often 200/400Hz on mobile phones), so we firstly do the procedure of $pre-integration$. Variables are represented in a local coordinate of frame k. This is the same with [11, 27]'s implementation:

$$\begin{aligned}{}^{I_k}\hat{\boldsymbol{\alpha}}_{i+1} &= {}^{I_k}\hat{\boldsymbol{\alpha}}_i + {}^{I_k}\hat{\boldsymbol{\beta}}_i\Delta t + \frac{1}{2}\boldsymbol{R}({}^{I_k}\hat{\boldsymbol{\gamma}}_i)(\hat{\boldsymbol{a}}_i - \hat{\boldsymbol{b}}_{a_k})\Delta t^2 \\ {}^{I_k}\hat{\boldsymbol{\beta}}_{i+1} &= {}^{I_k}\hat{\boldsymbol{\beta}}_i + \boldsymbol{R}({}^{I_k}\boldsymbol{\gamma}_i)(\hat{\boldsymbol{a}}_i - \hat{\boldsymbol{b}}_{a_k})\Delta t \\ {}^{I_k}\hat{\boldsymbol{\gamma}}_{i+1} &= {}^{I_k}\hat{\boldsymbol{\gamma}}_i Quat((\hat{\boldsymbol{w}}_i - \hat{\boldsymbol{b}}_{g_k})\Delta t)\end{aligned} \tag{10}$$

Here, we donate $Quat(\cdot)$ as the function to transpose a rotation from $Euler\ Angles$ to $Quaternion$. And then we further define the cost term of IMU measurements based (10):

$$\begin{aligned}C_{u_k} &= ||\begin{bmatrix} {}^{I_k}\boldsymbol{R}_W({}^{W}\boldsymbol{p}_{I_{k+1}} - {}^{W}\boldsymbol{p}_{I_k} + \frac{1}{2}{}^{W}\boldsymbol{g}\Delta t_k^2 - {}^{W}\boldsymbol{v}_{I_k}\Delta t_k) - {}^{I_k}\hat{\boldsymbol{\alpha}}_{I_{k+1}} \\ {}^{I_k}\boldsymbol{R}_W({}^{W}\boldsymbol{v}_{I_{k+1}} + {}^{W}\boldsymbol{g}\Delta t_k - {}^{W}\boldsymbol{v}_{I_k}) - {}^{I_k}\hat{\boldsymbol{\beta}}_{I_{k+1}} \\ 2[({}^{W}\boldsymbol{q}_{I_k})^{-1} \otimes {}^{W}\boldsymbol{q}_{I_{k+1}} \otimes ({}^{I_k}\hat{\boldsymbol{\gamma}}_{I_{k+1}})^{-1}]_{xyz} \\ \boldsymbol{b}_{a_{k+1}} - \boldsymbol{b}_{a_k} \\ \boldsymbol{b}_{g_{k+1}} - \boldsymbol{b}_{g_k} \end{bmatrix}||^2_{\boldsymbol{\Sigma}_u} \\ &= ||\boldsymbol{H}_{I_k}\tilde{\boldsymbol{S}}_{I_k} + \boldsymbol{H}_{I_{k+1}}\tilde{\boldsymbol{S}}_{I_{k+1}} + \boldsymbol{H}_{E_{k+1}}\tilde{\boldsymbol{S}}_{E_{k+1}} - \boldsymbol{r}_{u_{k+1}}||^2_{\boldsymbol{\Sigma}_u}\end{aligned} \tag{11}$$

where $[{}^{I_k}\hat{\boldsymbol{\alpha}}_{I_{k+1}}, {}^{I_k}\hat{\boldsymbol{\beta}}_{I_{k+1}}, {}^{I_k}\hat{\boldsymbol{\gamma}}_{I_{k+1}}]^T$ are pre-integrate IMU measurement terms between image frame k and $k+1$. $\boldsymbol{\Sigma}_u$ is the pre-integration covariance matrix. $\boldsymbol{H}$ is the Jacobian matrix of the corresponding state, and $\boldsymbol{r}_{u_{k+1}}$ is the measurement residual. This cost term is the same as [27].

5.3 State Augmentation and Marginalization

The prior information obtained from the previous time step k is $C_k(\tilde{\boldsymbol{S}}_k^{\oplus})$. At each time step $k+1$, new state $\boldsymbol{S}_{I_{k+1}}$ and $\boldsymbol{S}_{E_{k+1}}$ are added into the current state vector $\boldsymbol{S}_k$ using the IMU measurement. To maintain constant computational complexity, we marginalize the oldest cloned state $\boldsymbol{S}_{I_{k-(m-1)}}$ and last IMU state $\boldsymbol{S}_{E_k}$. Following [35] , we reorder the prior matrix and then perform QR decomposition to eliminate the state that needs to be marginalized. In the end, the prior matrix of the system after marginalization only contains states $\tilde{\boldsymbol{S}}_{k+1} = [\boldsymbol{S}^T_{I_{k-(m-2)}}, ..., \boldsymbol{S}^T_{I_{k+1}}, \boldsymbol{S}^T_{E_{k+1}}]^T$ that are not marginalized. The new prior cost function is:

$$C_{k+1}(\tilde{\boldsymbol{S}}_{k+1}) = ||H_{k+1}\tilde{\boldsymbol{S}}_{k+1} - \boldsymbol{r}_{k+1}||^2 \tag{12}$$

5.4 Image Process and Measurement

Since commercial-grade mobile phones are often with low computing power, we use FAST [29] to detect feature points and KLT [31] to track them. Both FAST and KLT are simple and with low computational complexity, easy to run on mobile phones. Similar to most MSCKF-like methods, feature tracks which are out of sliding window or tracking lost will be triangulated first, and refined by bundle adjustment (BA). Parameters of 3D points will be projected into nullspace, and only camera and IMU states remain in state vector and will be updated like [35]. The cost term of visual features is defined by:

$$C_z(\tilde{\boldsymbol{S}}_{k+1}) = ||\boldsymbol{H}_z\tilde{\boldsymbol{S}}_{k+1} - \tilde{\boldsymbol{r}}_z||^2_{\Sigma_z} \tag{13}$$

where Σ_z is the Gaussian covariance matrix of feature measurement.

5.5 Neural IMU Network Measurement

The outputs of neural work are relative displacement $\boldsymbol{d}_{ij}$ and measurement uncertainty $\boldsymbol{u}_{ij}$ between time t_i and t_j. $\boldsymbol{d}_{ij}$ is expressed in the inertial coordinate system. There are two points needing special attention here. First, the learned movement pattern is yaw angle quivariant, that is, $\boldsymbol{d}_{ij}$ changes with the change of the yaw angle at time t_i. Second, t_i and t_j are not aligned in time with the estimated state of the current sliding window. Therefore, to completely define the motion mode constraints, the propagation of IMU is needed. Assuming that the rotation matrix at time t_i is ${}^{W}R_{I_i}$, we use the Euler angle form of the external parameter XYZ to decompose the rotation matrix, that is, ${}^{W}R_{I_i} = R_\gamma R_\beta R_\alpha$, α, β, and γ respectively represent roll, pitch, and yaw angles. When the pitch angle is close to 90°, the Euler angle will have a deadlock phenomenon, then this conversion is invalid. Unlike [21], the constraint is directly discarded. When $|pitch - 90°| < 10°$, we use the external parameter "YXZ" conversion instead, that is, ${}^{W}R_{I_i} = R_\gamma R_\alpha R_\beta$ to avoid deadlock. Then the constraints provided by the network are:

$$\boldsymbol{C}_n(\tilde{\boldsymbol{S}}_{k+1}) = ||(R_\gamma)^T({}^{W}\boldsymbol{p}_{I_j} - {}^{W}\boldsymbol{p}_{I_i}) - \boldsymbol{d}_{ij}||^2_{\Sigma_n} \tag{14}$$

where ${}^{W}\boldsymbol{p}_{I_i}$, ${}^{W}\boldsymbol{p}_{I_j}$ are the corresponding positions in the world frame at time t_i, t_j, and Σ_n is the covariance matrix of this constraint. Assuming that the states in the window closest to time t_i and t_j are t_m and t_n, respectively. Through the pre-integration theory of IMU [11], the following constraints can be established:

$$\begin{aligned}{}^{W}\boldsymbol{p}_{I_i} &= {}^{W}\boldsymbol{p}_{I_m} + {}^{W}\boldsymbol{v}_{I_m}\Delta t_m - \frac{1}{2}{}^{W}\boldsymbol{g}\Delta t_m^2 + {}^{W}R_{I_m}\,{}^{I_m}\boldsymbol{\alpha}_{I_i} \\ {}^{W}\boldsymbol{p}_{I_j} &= {}^{W}\boldsymbol{p}_{I_n} + {}^{W}\boldsymbol{v}_{I_n}\Delta t_n - \frac{1}{2}{}^{W}\boldsymbol{g}\Delta t_n^2 + {}^{W}R_{I_n}\,{}^{I_n}\boldsymbol{\alpha}_{I_j} \\ {}^{W}R_{I_i} &= {}^{W}R_{I_m}\boldsymbol{R}({}^{I_m}\boldsymbol{\gamma}_{I_i})\end{aligned} \tag{15}$$

Here, ${}^{W}\boldsymbol{v}_{I_m}$ is the velocity of IMU in the world frame at time t_m. Δt_m equals t_m minus t_i. ${}^{I_m}\boldsymbol{\alpha}_{I_i}$, ${}^{I_n}\boldsymbol{\alpha}_{I_j}$, and ${}^{I_m}\boldsymbol{\gamma}_{I_i}$ is the mean part of IMU pre-integration [11].

Combining Equations (14) and (15), we can write the final constraint:

$$\begin{aligned}\boldsymbol{C}_n(\tilde{\boldsymbol{S}}_{k+1}) &= ||g({}^{W}R_{I_m}\boldsymbol{R}({}^{I_m}\boldsymbol{\gamma}_{I_i}))^T(({}^{W}\boldsymbol{p}_{I_n} + {}^{W}\boldsymbol{v}_{I_n}\Delta t_n - \frac{1}{2}{}^{W}\boldsymbol{g}\Delta t_n^2 + {}^{W}R_{I_n}\,{}^{I_n}\boldsymbol{\alpha}_{I_j}) \\ &\quad - ({}^{W}\boldsymbol{p}_{I_m} + {}^{W}\boldsymbol{v}_{I_m}\Delta t_m - \frac{1}{2}{}^{W}\boldsymbol{g}\Delta t_m^2 + {}^{W}R_{I_m}\,{}^{I_m}\boldsymbol{\alpha}_{I_i})) - \boldsymbol{d}_{ij}||^2_{\Sigma_n} \\ &= ||\boldsymbol{H}_n\tilde{\boldsymbol{S}}_{k+1} - \boldsymbol{r}_n||^2_{\Sigma_n}\end{aligned} \tag{16}$$

where $g(\cdot)$ is the function converting the rotation matrix to the yaw angle. The covariance of the rotating part is mainly related to the bias and noise of the gyroscope. In a short time, the noise of the gyroscope is much smaller than the noise of the accelerometer, and ${}^{I_n}\alpha_{I_j}$ has considered the noise of the gyroscope. To facilitate the calculation of covariance, we omit the covariance generated by the term $g({}^{W}R_m\,{}^{I_m}\gamma_{I_i})^T$. Then the final covariance of the constraint is:

$$\begin{aligned}\boldsymbol{\Sigma}_n = g({}^{W}R_{I_m}\,{}^{I_m}\boldsymbol{\gamma}_{I_i})^T({}^{W}R_{I_n}\,{}^{I_n}_{\alpha}\boldsymbol{P}_{I_j}({}^{W}R_{I_n})^T + {}^{W}R_{I_m}\,{}^{I_n}_{\alpha}\boldsymbol{P}_{I_i}({}^{W}R_{I_m})^T) \\ g({}^{W}R_{I_m}\,{}^{I_m}\boldsymbol{\gamma}_i) + \boldsymbol{\Sigma}_{ij}\end{aligned} \tag{17}$$

where ${}^{I_n}_{\alpha}P_{I_j}$ is the pose part of IMU pre-integration covariance matrix [11]. $\hat{\boldsymbol{\Sigma}}_{ij}$ is the covariance matrix of network output.

5.6 Outliers Removal

Before employing visual measurement and NIN measurement updates, we requires an outlier removal module to ensure the fusion system to be robust. Ours system employs the standard Mahalanobis distance test to protect the filter estimate. We discard the measurement when Mahalanobis distance $||\boldsymbol{d}-\hat{\boldsymbol{d}}||_{\boldsymbol{HPH}^T+\hat{\boldsymbol{\Sigma}}}$ is greater than a threshold, where H, P represent measurement Jacobian and corresponding covariance of state. For NIN measurement, we scale the covariance of network output by a factor of 10 to compensate for the inaccurate measurement.

5.7 State Update

Finally, the states are updated by minimizing all the measurements and prior terms. We stack all the constraints to get the following equation:

$$\min_{\tilde{\boldsymbol{S}}_{k+1}} C^{\oplus}_{k+1}(\tilde{\boldsymbol{S}}_{k+1}) = \min_{\tilde{\boldsymbol{S}}_{k+1}} \left\| \begin{bmatrix} \boldsymbol{H}_{k+1} \\ \Sigma_z^{-\frac{1}{2}} \boldsymbol{H}_z \\ \Sigma_n^{-\frac{1}{2}} \boldsymbol{H}_n \end{bmatrix} \tilde{\boldsymbol{S}}_{k+1} - \begin{bmatrix} \boldsymbol{r}_{k+1} \\ \Sigma_z^{-\frac{1}{2}} \boldsymbol{r}_z \\ \Sigma_n^{-\frac{1}{2}} \boldsymbol{r}_n \end{bmatrix} \right\|^2 \tag{18}$$

There are many methods to solve (18), here we adopt [22, 35]'s QR factorization to solve it. After that, we can get the estimation of $\tilde{\boldsymbol{S}}_{k+1}$. Then the state will be updated with the following equation:

$$\tilde{\boldsymbol{S}}^{\oplus}_{k+1} = \hat{\boldsymbol{S}}_{k+1} + \tilde{\boldsymbol{S}}_{k+1} \tag{19}$$

where $\hat{\boldsymbol{S}}_{k+1}$ includes the states estimated in the previous time step and new state from IMU propagation. Here, all the constraints are used for filter update, and QR factorized equation of (18) will serve as the new prior information for the next time step.

5.8 Initialization

A high-quality initialization will help speed up the filter convergence. We use different initialization methods according to different motion states, such as motion and static. If the average displacement of the sparse features of 3 consecutive frames of images on the image plane is lower than a certain threshold and the standard deviation of acceleration and angular velocity is lower than a certain threshold, we consider that it is currently in a stationary state, otherwise, it is moving. When it is currently in a static state, we try to perform static initialization. The initial translation is set to zero. The initial local gravity is the average of the accelerometer measurements between the latest two frames of images, and the initial rotation is aligned with the local initial gravity. The initial gyroscope bias is the average of gyroscope measurements between the latest two frames of images. The initial acceleration bias is set to zero. For motion initialization, we refer to this work [27]. We perform vision-only SfM first to get a pose without scale, and then align the IMU pre-integration results with the SfM results to recover the scale, velocity, gravity, and IMU bias.

6 Experiments

In this section, we conduct a series of experiments to verify the accuracy and generalization of our proposed neural inertial network (NIN), and the accuracy and robustness of our filter system, which fuses visual, inertial, and NIN information. Finally, we prove that our system has better practical application value through an online AR demo.

6.1 Baselines

We compare our method against the following baseline algorithms:

- **3D-RoNIN** [14]: a data-driven method, where the positions are estimated from a sequence of IMU sensor measurements. We evaluate all version of RoNIN (LSTM, TCN and ResNet) using their open source implementation.
- **TLIO** [21]: the tightly-coupled method, fuse the measurement from trained network and kinematic motion model into a stochastic cloning EKF to solve for pose, velocity and IMU biases.
- **VINS-Mono** [27]: a representative state-of-the-art visual inertial odometry with open source code.
- **ARCore**: a representative open platform for building augmented reality apps on Android. Their motion tracking technology uses a fusion of visual and inertial data allowing the phone to understand and track its position relative to the world.

We note that baseline methods can be divided into two categories: **1) Pose estimates based on IMU data (3D-RoNIN and TLIO)**. We compared with these methods to demonstrate the superiority of our IMU neural network on open-source IDOL dataset [33]. These competing methods and our proposed one estimate the translation assuming the rotation is known. So we use the ground truth orientation to estimate the translation for all methods. We use the same training and validation data with the baseline methods to make a fair comparison. Since the open-source IDOL [33] has not yet been announced, and the article does not explain how to divide the training data, we do not compare our method with it. **2) Pose estimates using IMU and visual sensor fusion (VINS, ARCore)**. These methods are used to demonstrate that visual inertial navigation algorithmm, fusing with human motion model from neural network, can obtain higher accuracy and robustness.

6.2 Metrics Definitions

To demonstrate the effectiveness of our system, Absolute Trajectory Error (ATE), Time-Relative Trajectory Error(T-RTE), and Distance-Relative Trajectory Error (D-RTE) metrics introduced in [33] is adopted. For the sequence without ground-truth, we define the Final Translation Drift (FTD)(%) metric as

$$\|\hat{\boldsymbol{p}}_e - \hat{\boldsymbol{p}}_s\| / L.$$

If the starting point and ending point of the sequence are roughly the same, we can use FTD to measure position accuracy without ground-truth. $\hat{\boldsymbol{p}}_e, \hat{\boldsymbol{p}}_s$ are the starting and ending positions estimated by the algorithm, and L is the trajectory length.

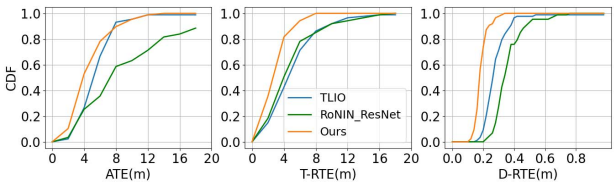
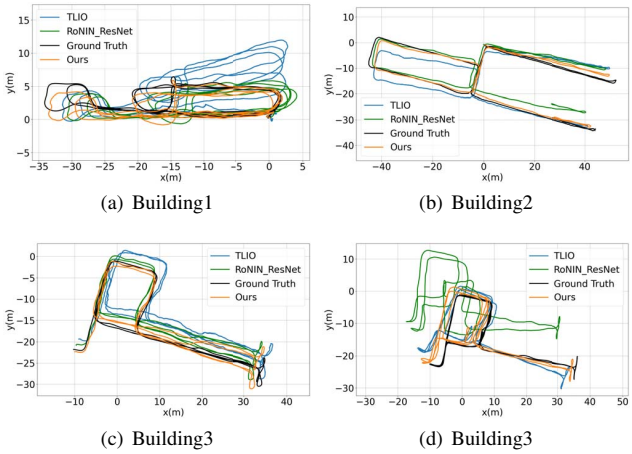


Fig. 4: The cumulative density function of corresponding metrics on the entire test set.

6.3 Neural Inertial Network Performance

IDOL datasets: The dataset has been divided into 3 buildings. Each building dataset is grouped into two subsets: 'known' and 'unknown'. The users of the training set are the same as the 'known' data, and the training set only includes building 1. This allows



(a) Building1 (b) Building2

(c) Building3 (d) Building3

Fig. 5: Trajectories of different methods on IDOL dataset. Each trajectory have same starting point.

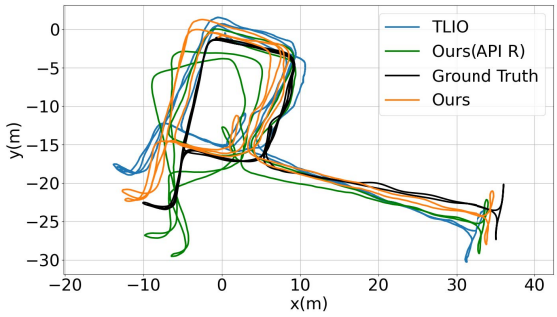


Fig. 6: Trajectories using different orientation.

for testing the generalization of networks across users, buildings and devices. We use the same training and validation data with the corresponding baseline methods to make a fair comparison. Since the neural inertial network estimates relative displacement in the known gravity-aligned frame, during the test, we mainly verify the performance of the network to estimate the displacement under the ground-truth gravity direction.

Fig. 4 shows the error distribution on the entire test set. It confirmed that compared with RoNIN and TLIO, our system consistently maintains higher accuracy.

Tables 1 and 2 show that the IMU neural network can be generalized to different people, different devices, and different environments. TLIO (wo EKF) is the TLIO pure network version without EKF framework. Ours (wo AL) is our proposed network without absolute loss. Our network has obvious advantages over other baselines in all metrics. Our end-to-end network does not need to use EKF fusion like TLIO, and can also achieve the best local and global

Table 1: ATE metrics on IDOL dataset.

Model	Building 1		Building 2		Building 3	
	Known	Unknown	Known	Unknown	Known	Unknown
RoNIN-LSTM	18.33	15.18	45.58	45.58	32.68	25.81
RoNIN-TCN	25.81	25.81	41.42	35.83	35.17	43.99
RoNIN-ResNet	2.90	4.46	15.27	11.82	10.03	13.74
TLIO(wo EKF)	5.58	6.08	7.23	7.09	7.04	6.30
TLIO	3.75	4.14	6.23	7.17	5.06	5.23
Ours (wo AL)	3.43	4.78	9.53	7.41	9.77	9.65
Ours	**2.71**	**3.62**	**6.19**	**5.23**	**4.57**	**3.38**

Table 2: D-RTE, T-RTE metrics on IDOL dataset.

Model	Building 1		Building 2		Building 3	
	T-RTE	D-RTE	T-RTE	D-RTE	T-RTE	D-RTE
RoNIN-LSTM	6.77	0.66	15.58	1.06	9.08	0.73
RoNIN-TCN	8.61	0.77	19.14	0.985	11.33	0.85
RoNIN-ResNet	2.08	0.32	7.58	0.36	4.73	0.4
TLIO(wo EKF)	2.05	0.21	5.00	0.22	5.32	0.22
TLIO	2.52	0.28	7.25	0.26	4.94	0.29
Ours (wo AL)	1.72	0.25	5.05	0.23	3.60	0.24
Ours	**1.49**	**0.17**	**4.19**	**0.19**	**2.99**	**0.20**

Table 3: Metrics using different orientation.

Model	API Orientation			Ground Truth Orientation		
	ATE	T-RTE	D-RTE	ATE	T-RTE	D-RTE
RoNIN-ResNet	6.88	7.10	0.42	9.75	4.80	0.36
TLIO(wo EKF)	8.82	7.36	0.35	6.55	4.12	0.22
TLIO	**5.27**	**5.16**	**0.27**	5.26	4.9	0.28
Ours	6.53	6.75	0.32	**4.28**	**2.89**	**0.19**

accuracy. Human movement has a certain rhythm and regularity. We recognize human movement characteristics through ResNet, and then use LSTM to learn the regularity of movement in time series. Therefore, compared with the motion recognition of a single window, we use historical motion features to achieve better local relative accuracy and global accuracy. The relative loss and absolute loss can make the network care about the local accuracy and also pay attention to the long-term global accuracy. The tables also confirm the effectiveness of the loss. Fig. 7 shows the trajectories of different methods on IDOL sequence.

Table 3 shows the displacement accuracy of the inertial network under different rotation inputs. The accuracy of RoNIN, TLIO (wo EKF) and our network decreases with the decrease of rotation accuracy, while TLIO is not affected because of its own estimation of rotation. The visual-inertial system can estimate the reliable gravity direction, which will help improve the accuracy of the inertial network. Fig. 6 confirms that low-precision rotation input will increase the yaw direction error, which leads to a decrease in the overall trajectory accuracy.

Our Data: Table 4 shows the performance comparison on our self-collected data. Our system achieves the best local and global accuracy, and the conclusion is consistent with IDOL datasets.

6.4 EKF Fusion System Performance

We use Huawei Mate20 Pro mobile phone to collect the verification data in indoor scenes and outdoor scenes for this subsection. Indoor scenes are equipped with VICON equipment, which can give ground-truth trajectories to measure the accuracy of the recovered trajectories. In outdoor scenes, the starting point and ending points of each sequence are roughly the same, so we choose the error between the starting point and the ending point to measure the positioning errors. We also split the data set into normal data and challenging data. Normal data includes a high-quality visual environment, such as scene depth in the range of 1-5m, good lighting conditions, no dynamic obstacles, no occlusion, no image blur, etc. The opposite is challenging data.

Table 4: Comparison on Our Data.

Model	Ourdoor			Indoor		
	ATE	T-RTE	D-RTE	ATE	T-RTE	D-RTE
RoNIN-LSTM	16.62	15.47	1.23	2.41	1.62	0.55
RoNIN-TCN	18.33	17.01	1.33	2.72	1.81	0.57
RoNIN-ResNet	1.93	2.19	0.24	1.49	**1.33**	0.58
TLIO	1.82	2.06	**0.15**	2.09	1.73	**0.30**
Ours	**1.55**	**1.81**	**0.15**	**1.24**	1.43	**0.30**

We note that our RNIN model used in this experiment is trained on our own collected data, which is only about 7 hours with not very accurate ground truth. However experiment results show our fusion system still performs quite well.

It can be seen from the Table 5 that the accuracy of RNIN-VIO is close to that of BVIO in normal scenarios. The traditional visual-inertial system does not reduce the accuracy of the system due to the integration of low-precision NIN observations. In challenging scenarios, the accuracy of RNIN-VIO is significantly better than that of VIO. Therefore, the NIN fusion system can maintain more robust tracking in extreme scenarios. VINS-Mono performed poorly on our dataset. The possible reason is that it does not consider dynamic exposure time and we use the default parameters.

Table 6 shows the final translation error of the outdoor. Each outdoor sequence is about 430m in length and is mainly divided into normal scenes and challenging scenes. Challenge scenes include occlusion, distant view, normal walking and shaking, dynamic pedestrians, and weak textures. All Challenging means that all challenges are included in the entire sequence. The outdoor test results are similar to the indoor results. In normal scenarios, the accuracy of the NIN-integrated VIO algorithm is similar to that of the traditional VIO algorithm, and the fusion vision algorithm has higher accuracy than the pure IMU methods, which is reasonable because vision can provide a rich map of the external environment to greatly reduce the cumulative error of the system. In challenging scenarios, the NIN method has better robustness. The VIO algorithm fused with NIN has higher robustness and higher accuracy. Among them, the accuracy of two sequences has a certain decrease, mainly due to the NIN measurements outlier. How to accurately eliminate the outliers of NIN will be our next work.

Table 5: ATE (m) on Indoor room with VICON equipment.

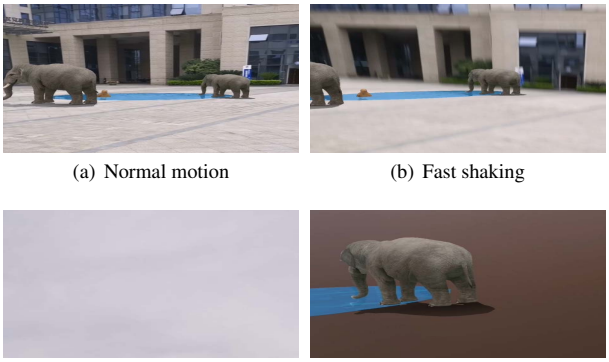
Sequence	VINS-Mono	BVIO	RNIN-VIO
Normal00	0.256	0.146	**0.124**
Normal01	0.362	0.078	**0.077**
Normal02	0.228	0.081	**0.078**
Normal03	0.418	**0.150**	0.157
Normal04	2.766	0.215	**0.212**
Normal05	0.415	0.383	**0.318**
Challenging00	X	3.553	**3.150**
Challenging01	5.452	1.563	**1.318**
Challenging02	X	0.743	**0.520**

Table 6: FTD(%) on ourdoor.

Sequence	VINS-Mono	BVIO	RNIN	RNIN-VIO
Normal00	8.81	1.95	2.67	**1.81**
Normal01	X	1.89	5.35	**1.65**
Occlusion00	X	X	3.88	**2.27**
Occlusion01	X	8.57	4.14	**2.08**
Far Scenes00	X	3.62	4.78	**3.60**
Far Scenes01	X	X	5.22	**3.34**
Shake00	X	4.50	5.59	**3.80**
Shake01	X	**2.15**	7.29	6.00
All Challenging00	X	X	7.23	**3.14**
All Challenging01	X	27.3	2.17	**1.32**

6.5 Augmented Reality Application

We have developed an Android application to demonstrate the RNIN-VIO's rubustness and accuracy. We use tensorflow [1] to do the inference of our NIN. Other part of the system fully run on CPU platform. We take Huawei's Mate20 pro as the testing device. The input are 30Hz images with 512×384 resolution and 200 Hz IMU data. The module of NIN run in a background thread with 2Hz frame rate, and cost time is 43ms. VIO module has a high processing speed, and the processing time is much less than the input image frames' interval.



(a) Normal motion (b) Fast shaking

(c) Looking at far sky (d) Camera occlusion (with hands)

Fig. 7: **Mobile AR:** We test RNIN-VIO in different types of challenging situations, including fast shaking & motion blur, looking at far sky, and camera occlusion for a long time. RNIN-VIO works well in all these challenging situations.

We add some virtual 3D animals into the scene in our AR demo. We evaluate the robustness in different challenging situations, such as fast shaking with motion blur, looking at far sky, and camera occlusion for a long time. In all of these situations, RNIN-VIO achieves a robust and continuous 6DoF tracking without tracking lost. Please watch the supplementary video for the complete results.

7 Generalizability

The test set includes unknow sequences (such as different users, devices and environments) that are not included in the training set in subsection 6.3. Tables 1, 2 and 4 show that the IMU neural network can be generalized to different people, different devices, and different environments. We believe that this generalization is mainly due to the similar acceleration and angular velocity distribution of different people's movements, the noise of different equipment can be simulated through data enhancement, and the IMU measurements have nothing to do with the external environment.

In the experiment, we also found that our system will fail for the types of motion that do not appear in the training set (such as very slow walking, sitting on a chair and moving, drone movement, etc.). Like general supervised learning algorithms, our system has difficulties in accurately estimating the movements that are not included in the training set. How to adapt the IMU neural network to various motions through online learning will be our future work.

8 Conclusion

In this article, we propose a tightly coupled visual-inertial odometry with the help of NIN. Our objective is to enhance the robustness and accuracy of traditional visual-inertial systems by using highly robust but relative low precision NIN information. We design a robust IMU neural network to learn human movement priors from IMU data. The experiments prove that our network has reached the level of state-of-the-art. Then we further fuse NIN and VI information and improve the robustness of odometry in extremely challenging environments while maintaining the original high-precision level. The online AR demo also proves the application potential of our system in mobile augmented reality.

Acknowledgments

The authors are very grateful to Shangjin Zhai, Chongshan Sheng, Yuequ Cai, and Kai Sun for their kind help in developing RNIN-VIO system. This work was partially supported by NSF of China (Nos. 61822310 and 61932003).

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct EKF-based approach. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 298–304. IEEE, 2015.

[3] A. Brajdic and R. Harle. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 225–234, 2013.

[4] M. Brossard, A. Barrau, and S. Bonnabel. AI-IMU dead-reckoning. *IEEE Transactions on Intelligent Vehicles*, 5(4):585–595, 2020.

[5] M. Brossard, S. Bonnabel, and A. Barrau. Denoising IMU gyroscopes with deep learning for open-loop attitude estimation. *IEEE Robotics and Automation Letters*, 5(3):4796–4803, 2020.

[6] C. Chen, X. Lu, A. Markham, and N. Trigoni. IONet: Learning to cure the curse of drift in inertial odometry. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[7] Q. A. Dugne-Hennequin, H. Uchiyama, and J. P. S. D. M. Lima. Understanding the behavior of data-driven inertial odometry with kinematics-mimicking deep neural network. *IEEE Access*, 9:36589–36619, 2021.

[8] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2017.

[9] M. A. Esfahani, H. Wang, K. Wu, and S. Yuan. AbolDeepIO: A novel deep inertial odometry network for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):1941–1950, 2019.

[10] M. A. Esfahani, H. Wang, K. Wu, and S. Yuan. OriNet: Robust 3-D orientation estimation with a single particular IMU. *IEEE Robotics and Automation Letters*, 5(2):399–406, 2019.

[11] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. Georgia Institute of Technology, 2015.

[12] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15–22. IEEE, 2014.

[13] E. Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, 2005.

[14] S. Herath, H. Yan, and Y. Furukawa. RoNIN: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3146–3152. IEEE, 2020.

[15] A. R. Jiménez, F. Seco, J. C. Prieto, and J. Guevara. Indoor pedestrian navigation using an INS/EKF framework for yaw drift reduction and a foot-mounted IMU. In *Proceedings of 7th Workshop on Positioning, Navigation and Communication*, pp. 135–143. IEEE, 2010.

[16] W. Kang and Y. Han. SmartPDR: Smartphone-based pedestrian dead reckoning for indoor localization. *IEEE Sensors Journal*, 15(5):2906–2916, 2014.

[17] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234. IEEE, 2007.

[18] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[19] M. Li and A. I. Mourikis. Improving the accuracy of EKF-based visual-inertial odometry. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 828–835. IEEE, 2012.

[20] H. Liu, G. Zhang, and H. Bao. Robust keyframe-based monocular SLAM for augmented reality. In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 1–10. IEEE, 2016.

[21] W. Liu, D. Caruso, E. Ilg, J. Dong, A. I. Mourikis, K. Daniilidis, V. Kumar, and J. Engel. TLIO: Tight learned inertial odometry. *IEEE Robotics and Automation Letters*, 5(4):5653–5660, 2020.

[22] P. S. Maybeck. *Stochastic models, estimation, and control*. Academic press, 1982.

[23] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 3565–3572. IEEE, 2007.

[24] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[25] U. Onyekpe, V. Palade, and S. Kanarachos. Learning to localise automated vehicles in challenging environments using inertial navigation systems (INS). *Applied Sciences*, 11(3):1270, 2021.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

[27] T. Qin, P. Li, and S. Shen. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[28] S. Rajagopal. Personal dead reckoning system with shoe mounted inertial sensors. *Master's Degree Project, Stockholm, Sweden*, 2008.

[29] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of European Conference on Computer Vision*, vol. 1, pp. 430–443, May 2006. doi: 10.1007/11744023_34

[30] R. L. Russell and C. Reale. Multivariate uncertainty in deep learning. *arXiv preprint arXiv:1910.14215*, 2019.

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

[32] J. P. Silva do Monte Lima, H. Uchiyama, and R.-i. Taniguchi. End-to-end learning framework for IMU-based 6-DoF odometry. *Sensors*, 19(17):3777, 2019.

[33] S. Sun, D. Melamed, and K. Kitani. IDOL: Inertial Deep Orientation-Estimation and Localization. *arXiv preprint arXiv:2102.04024*, 2021.

[34] D. Weber, C. Gühmann, and T. Seel. Neural networks versus conventional filters for Inertial-Sensor-based attitude estimation. In *Proceedings of IEEE 23rd International Conference on Information Fusion (FUSION)*, pp. 1–8. IEEE, 2020.

[35] K. Wu, A. Ahmed, G. A. Georgiou, and S. I. Roumeliotis. A square root inverse filter for efficient vision-aided inertial navigation on mobile devices. In *Robotics: Science and Systems*, vol. 2, 2015.