Learning to Compensate for the Drift and Error of Gyroscope in Vehicle Localization

Xiangrui Zhao[1], Chunfang Deng[1], Xin Kong[1], Jinhong Xu[1], and Yong Liu[1,2]

***Abstract*— Self-localization is an essential technology for autonomous vehicles. Building robust odometry in a GPS-denied environment is still challenging, especially when LiDAR and camera are uninformative. In this paper, We propose a learning-based approach to cure the drift of gyroscope for vehicle localization. For consumer-level MEMS gyroscope (stability 10), our *GyroNet* can estimate the error of each measurement. For high-precision Fiber optics Gyroscope (stability 0.05), we build a *FoGNet* which can obtain its drift by observing data in a long time window. We perform comparative experiments on publicly available datasets. The results demonstrate that our *GyroNet* can get higher precision angular velocity than traditional digital filters and static initialization methods. In the vehicle localization, the *FoGNet* can effectively correct the small drift of the Fiber optics Gyroscope (FoG) and can achieve better results than the state-of-the-art method.**

I. Introduction

For ground vehicles, localization in the environment is the basis for path planning and motion control. In an outdoor open environment, the vehicle can use the GPS to obtain an accurate position. However, in the mountainous environment of the wild and the urban environment with many tall buildings, GPS signals are often obscured, and the localization accuracy drops sharply. Although we can use visual odometry and laser odometry as complements to achieve high localization accuracy without GPS, they will become unstable due to the poor illumination, snowy or foggy scenes.

Compared to visual odometry and laser odometry, wheel odometry has little dependence on the environment. It still works when the camera and LiDAR are uninformative. Vehicles have wheel odometers that measure the speed of the wheels and then calculate bodies' linear velocity and angular velocity. Due to noisy measurement, the traditional methods usually use the linear velocity only and acquire the angular velocity by the gyroscope. Although the gyroscope's error is smaller than that calculated by wheel odometers, its drift and error still harm the localization. Traditional denoising methods, including digital filtering, static initialization, and multi-sensor fusion, have developed for many years, but they don't perform well in vehicle localization. In this paper, we propose a learning-based approach to compensate for the gyroscopes with various precision. Since the MEMS gyroscopes and Fiber optics Gyroscopes have different work principles, their noise properties are diverse. We use two different networks for them to reduce the complexity of the model. Different from the end-to-end method of IONet [1], we only use neural networks to eliminate the drift and error of the sensor, while the localization still follows the traditional way in section III.A. The advantage of our approach is that it can achieve excellent results with a relatively simple neural network. It is also easy to train and more interpretable. Our contributions can be concluded as follows:

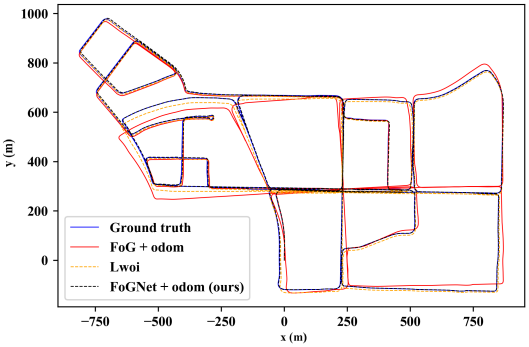


Fig. 1. The trajectory of Urban09.

- For noisy gyroscope in the low-cost MEMS IMU, we propose a *GyroNet* to estimate the error of each measurement and compensate.
- For high-precision Fiber optics Gyroscope (FoG), we propose a *FoGNet* to estimate the drift by observing sensor measurements in a long time window.
- We also evaluate our approach on publicly available datasets. The experiment results demonstrate that our approach outperforms the existing state-of-the-art method $lwoi$ [2].

II. Related Work

Odometry technology has made significant progress in recent years. IMU is first used in inertial navigation systems, which relies on linear acceleration and angular velocity measurements to estimate the pose of IMU. The most mature method is zero-velocity aided INS using a range of threshold-based detection methods [3]–[5]. When detecting a zero-velocity event, the velocity state can be fused with the dead reckoning motion model in an extended Kalman filter [6] or Bayesian filter. However, due to the severe error and long-term drift, traditional INS can hardly get accurate pose estimation using low-cost IMU. To improve localization accuracy, there are methods of fusing IMU measurements with image information. Filter-based methods like MSCKF [7], [8] add the image feature to the state vector together with

[1]Xiangrui Zhao ,Chunfang Deng, Xin Kong and Jinhong Xu are with the Institute of Cyber-Systems and Control, Zhejiang University, Zhejiang, 310027, China.

[2]Yong Liu is with the State Key Laboratory of Industrial Control Technology and Institute of Cyber-Systems and Control, Zhejiang University, Zhejiang, 310027, China (Yong Liu is the corresponding author, email: yongliu@iipc.zju.edu.cn).

IMU. They perform well but are sensitive to IMU noise parameters. VINS Mono [9] is a robust optimization-based method that needs a specific initialization process to estimate the bias of gyroscope and accelerometer. When the motion excitation is insufficient during initialization, it usually fails. VINS on wheels [10] adds a planar-motion constraint using wheel odometers to improve localization accuracy.

Deep learning methods are introduced into the odometry estimation. VINet [11] is an end-to-end network for processing images and IMU data. It outperforms some traditional methods while it's hard to train. IONet [1] is an end-to-end inertial odometry based on IMU only. It's robust to various environments but not accurate enough. Although these end-to-end methods work, they are not the best solution since the search space of the neural network is very large and is easy to be over-fitting or under-fitting. Many works combine deep learning with traditional methods. Wagstaff [12] uses LSTM for zero-velocity detection in inertial navigation. DeepVIO [13] has an IMU status update scheme to update the additional gyroscope and accelerometer bias. RINS-W [14] builds an LSTM-based motion profile detector and localizes through invariant Kalman filter. Lwoi [2] uses Gaussian Process and variational inference to estimate wheel odometry and IMU errors.

III. Our Learning-based Approach

Our learning-based approach consists of two neural networks on different categories of gyroscopes. We propose a *GyroNet* for noisy gyroscope in the low-cost IMU. It estimates the bias and noise of each measurement by the IMU data in a short sliding window. For high-precision Fiber optics Gyroscope, we build a novel *FoGNet* that takes IMU, odometers, and yaw-axis FoG measurements in a long time window as input and generates bias and noise estimation. In this section, we will first present the formal model of the sensors and then give these two neural networks.

A. Sensor Model

1) IMU Model: IMU measures angular velocity $\tilde{\omega}^b$ and acceleration $\tilde{\mathbf{a}}^b$ in the body frame, which are given by:

$$\tilde{\omega}^b = \omega^b + \mathbf{b}^g + \mathbf{n}^g \tag{1}$$

$$\tilde{\mathbf{a}}^b = \mathbf{q}_{bw}\left(\mathbf{a}^w + \mathbf{g}^w\right) + \mathbf{b}^a + \mathbf{n}^a \tag{2}$$

where superscript w represents the world frame, and superscript b represents the body frame. The measurements are affected by gyroscope bias $\mathbf{b}^g$, accelerometer bias $\mathbf{b}^a$ and additive noise. We assume that the additive noise are zero-mean Gaussian noise, $\mathbf{n}^a \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\sigma}_a^2\right)$, $\mathbf{n}^g \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\sigma}_g^2\right)$. $\mathbf{g}^w$ is gravity in the world frame and $\mathbf{q}_{bw}$ is the rotation from the world frame to the body frame.

2) Odometry model: For a vehicle equiped with wheel odometers, FoG and IMU on the ground, we define its state at time *t* as:

$$\mathbf{x}_t = \left[x_t, y_t, \phi_t, \theta_t, \psi_t, p_t, q_t, r_t\right]^T \tag{3}$$

where $\left[x_t, y_t\right]^T$ is the vehicle's position in the local frame. $\left[\phi_t, \theta_t, \psi_t\right]^T$ are the Euler angles representing the vehicle's orientation, and $\left[p_t, q_t, r_t\right]^T$ are the angular velocity in body frame. Since the vehicle moves on a plane, the altitude is not observable by the wheel odometers and IMU.

For a given time step δt, the process model takes $\mathbf{u}_t = \left[v_r, v_l, \delta_\psi\right]^T$ as input, where v_r and v_l are the speed measured by right and left wheel odometer. $\delta\psi$ is the change in yaw axis. Based on planar-motion assumption, the state update equation is given by:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{z}_t \delta t + \mathbf{w}_t \tag{4}$$

where $\mathbf{z}_t = \left[v\cos\left(\psi_t\right), v\sin\left(\psi_t\right), 0, 0, \delta\psi/\delta t, 0, 0, 0\right]^T$ and $\mathbf{w}_t$ is a zero-mean Gaussian noise. v is the velocity of the vehicle equals the average of v_r and v_l.

$$\delta\psi/\delta t = \tilde{\omega}^b - c^g \tag{5}$$

Our learning-based approach is to estimate compensation parameters c^g through neural network, c_M^g for MEMS gyroscope and c_F^g for Fiber optics Gyroscope(FoG). These compensation parameters are not constant yaw rate bias. We think that they contain errors such as measurement noise, random walk, and other non-artificial modeling errors. Then we subtract them from the raw data to compensate and calculate the trajectory of the vehicle follow Eq. 4.

B. GyroNet

We are inspired by deep neural networks to estimate the bias and noise of gyroscope. By treating a sequence of IMU data as an image, it is possible to apply CNN on sensor data. We evaluate different network structures, including CNN, RNN, and LSTM, in the training process. Fig.2 gives the test error of different frameworks. It shows that Bi-LSTM can converge to a small error as fast as possible. Therefore, we adopt Bi-LSTM as our backbone.

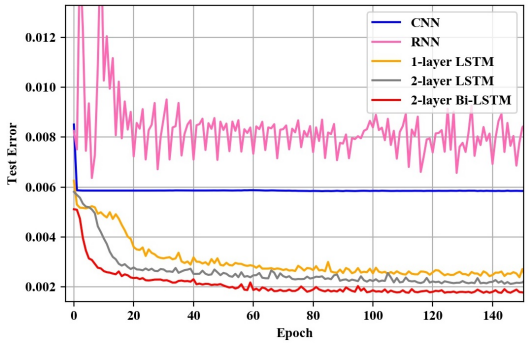


Fig. 2. Test error of adopting different framework.

LSTM is a special type of RNN that has long-term dependency. It combines the current input $\mathbf{x}(t)$ with the network's hidden state $\mathbf{h}(t-1)$:

$$\mathbf{h}(t) = \phi(\mathbf{W}_{hx}\mathbf{x}(t) + \mathbf{W}_{hh}\mathbf{h}(t-1)). \tag{6}$$

The output $\mathbf{h}(t)$ is the activation result of the current input and the previous hidden state. $\phi(\cdot)$ is an activation function. The matrices $\mathbf{W}_{hx}$ and $\mathbf{W}_{hh}$ are the weights that updated during training process.

In addition to the hidden state $\mathbf{h}(t)$, LSTM also propagates an internal state $\mathbf{s}(t)$. At each time, updates to $\mathbf{s}(t)$ contain two parts: the input gate $\mathbf{i}(t)$ and the forget gate $\mathbf{f}(t)$.

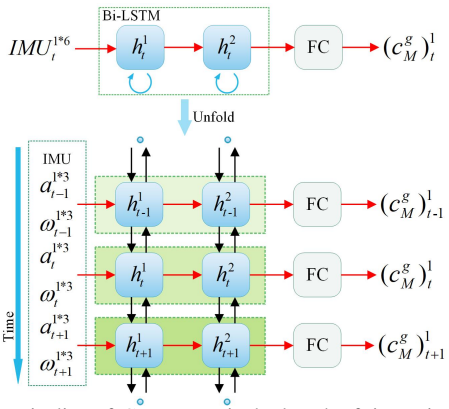


Fig. 3. The pipeline of *GyroNet*. *N* is the length of time window and *s* is the step size.

The input gate determines which part of the input $\mathbf{g}(t)$ is added to the state, while the forget gate removes the part of the hidden state that is no longer needed. The output gate $\mathbf{o}(t)$ is used to select the elements of $\mathbf{s}(t)$ that will be passed into $\mathbf{h}(t)$,

$$\mathbf{g}(t) = \phi(\mathbf{W}_{\mathrm{gx}}\mathbf{x}(t) + \mathbf{W}_{\mathrm{gh}}\mathbf{h}(t-1) + \mathbf{b}_{\mathrm{g}}), \quad (7)$$

$$\mathbf{i}(t) = \sigma(\mathbf{W}_{\mathrm{ix}}\mathbf{x}(t) + \mathbf{W}_{\mathrm{ih}}\mathbf{h}(t-1) + \mathbf{b}_{\mathrm{i}}), \quad (8)$$

$$\mathbf{f}(t) = \sigma(\mathbf{W}_{\mathrm{fx}}\mathbf{x}(t) + \mathbf{W}_{\mathrm{fh}}\mathbf{h}(t-1) + \mathbf{b}_{\mathrm{f}}), \quad (9)$$

$$\mathbf{o}(t) = \sigma(\mathbf{W}_{\mathrm{ox}}\mathbf{x}(t) + \mathbf{W}_{\mathrm{oh}}\mathbf{h}(t-1) + \mathbf{b}_{\mathrm{o}}), \quad (10)$$

$$\mathbf{s}(t) = \mathbf{g}(t) \odot \mathbf{i}(t) + \mathbf{s}(t-1) \odot \mathbf{f}(t), \quad (11)$$

$$\mathbf{h}(t) = \phi(\mathbf{s}(t)) \odot \mathbf{o}(t) \quad (12)$$

where $\sigma(\cdot)$ is the sigmoid activation function and $\odot$ represents multiplication of elements. The matrices $\mathbf{W}_{gx}$, $\mathbf{W}_{gh}$, $\mathbf{W}_{ix}$, $\mathbf{W}_{ih}$, $\mathbf{W}_{fx}$, $\mathbf{W}_{fh}$, $\mathbf{W}_{ox}$, $\mathbf{W}_{oh}$ are weight parameters. The biases $\mathbf{b}_g$, $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$ are trainable parameters of the network.

The LSTM only extracts information in the forward sequence and can not use the backward sequence, which explains why Bi-LSTM performs better in our experiment. Our *GyroNet* consists of a 2-layer Bi-LSTM with 60 units per layer. We include a single fully-connected layer after LSTM to reduce the network's output to 2D. The loss function is defined as the mean absolute error of prediction. The neural network works as a function f that maps sensor measurements to compensation parameters over a sliding window:

$$(\mathbf{a}, \omega)^{N*6} \xrightarrow{f} (c_M^g)^N \quad (13)$$

where $\mathbf{a}$ is acceleration, ω is angular velocity and N is window length.

C. FoGNet

Since the noise of low-cost MEMS gyroscope is very strong $(\sim 10^\circ/h)$, a simple network can work well. As shown in Tab.6, although the $GyroNet$ gets smaller final orientation error, it is still worse than FoG on average. For a high-precision FoG whose noise $(\sim 0.05^\circ/h)$ is 200 times smaller than the low-cost one, it requires a more complicated network.

Fig. 4 shows the pipeline of FoGNet. The input data is low-cost IMU, wheel odometers, and yaw-axis FoG measurements in a time window with a dimension of N*9. We

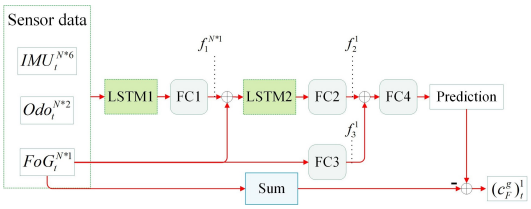


Fig. 4. The pipeline of *FoGNet*. N is the length of time window.

first transmit the input data into the *LSTM1*, a 2-layer Bi-LSTM with 90 units per layer, and obtain a N*1-dimensional vector f_1 through the fully-connected layer *FC1*. Then we combine f_1 with the measurements of the yaw-axis FoG from raw data, and transmit them to the *LSTM2*, a 2-layer Bi-LSTM with N units per layer. The layer *FC2* generates a scalar f_2. Next, we combine f_2 with f_3 generated by the fully-connected layer *FC3* and transmit it to the *FC4* to obtain a network prediction result. Finally, we subtract the FoG measurement from the prediction result and get the compensation parameter c_F^g.

The loss function consists of two parts. The first part is the Manhattan distance between the ground truth $\Delta\Theta$ and estimated value $\Delta\widetilde{\Theta}$ in a time window:

$$Loss_1 = \left\| \Delta\Theta - \Delta\widetilde{\Theta} \right\|_1 \quad (14)$$

We expect that the input data can get a preliminary denoising result after going through *LSTM1* and *FC1*, so f_1 should be close to the measurements of the FoG. To constrain the search space of the network parameters, we use the difference of f_1 and FoG measurements $\Delta\theta$ as a skip-connection loss.

$$Loss_2 = \left\| {f_1}^{N*1} - \Delta\theta^{N*1} \right\|_1 \quad (15)$$

The overall loss function is:

$$Loss = \left\| \Delta\Theta - \Delta\widetilde{\Theta} \right\|_1 + \alpha \left\| {f_1}^{N*1} - \Delta\theta^{N*1} \right\|_1 \quad (16)$$

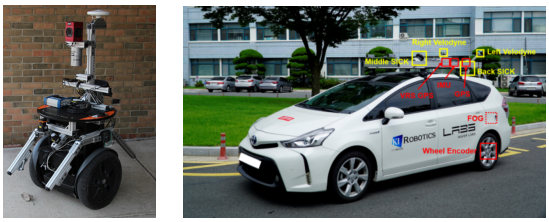
where α is a weight coefficient. We set it to 0.5.

IV. Experiments

We implement our model on the PyTorch framework and run the training process on an NVIDIA GTX 1080Ti GPU. During training, we used Adam, a first-order gradient-based optimizer with an initial learning rate of 0.0015.

A. GyroNet

1) Data Preparation: We evaluate our *GyroNet* on KAIST Urban Dataset [15]. The input data is low-cost gyroscope and accelerometer measurements in a time window. We collect training data through a sliding window with a



(a) Segway(NCLT) (b) Consumer car(KAIST Urban)

Fig. 5. Data acquisition platforms of two datasets.

TABLE I

ORIENTATION ERROR ON KAIST URBAN DATASET

	Mean error (rad)	Final error (rad)
Static initialization	0.001535	0.3153
FIR filter	0.002755	0.5146
Xsens internal filter	0.001662	0.2584
GyroNet	0.0003854	**0.06156**
FoG	**0.0003470**	0.06448

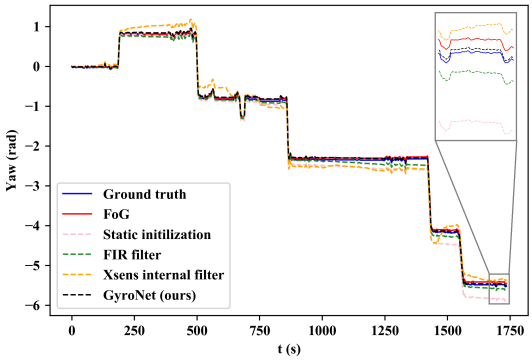


Fig. 6. The yaw angle of test sequence Urban14. Ground truth, FoG, and Xsens internal filter are provided directly from the dataset, and others are derived from angular velocity integration.

length of 0.1s and a step size of 0.01s. Then the data is shuffled to prevent over fitting. In this way, the model is not affected by the sequence length. The network prediction is the compensation parameters of the gyroscope getting from the difference of measurements and ground truth data.

2) Comparison: We compare 5 methods:

- **Static initialization:** performing a 3s initialization to estimate the bias of gyroscope.
- **FIR filter:** filtering the data by an FIR filter with $f_{pass} = 10Hz$ and $f_{stop} = 15Hz$.
- **Xsens internal filter:** the angle output of Xsens' internal multi-sensor fusion algorithm.
- **GyroNet (ours):** the integration of network denoising result.

In order to evaluate the performances of the above methods, we use the following two metrics:

- *Mean orientation error:* the average of orientation error on yaw with respect to FoG in 1s.
- *Final orientation error:* the final orientation error on yaw between the estimates and the FoG.

3) Results: We train our *GyroNet* on sequence `Urban06-13` and test on sequence `Urban14-17`. Tab. I shows that our approach has the smallest error compared with static initialization, FIR filter, and Xsens internal filter. The average final error even better than expensive high-precision FoG.

Furthermore, in Fig. 7, we calculate the vehicle trajectory follow Eq.4 using the corrected angular velocity and the wheel odometers, which makes it more intuitive to see how our approach corrects the sensor measurements.

B. FoGNet

1) Data Preparation: We evaluate our *FoGNet* on KAIST Urban Dataset [15] and NCLT Dataset [16]. The input data is low-cost gyroscope, accelerometer, and yaw-axis FoG

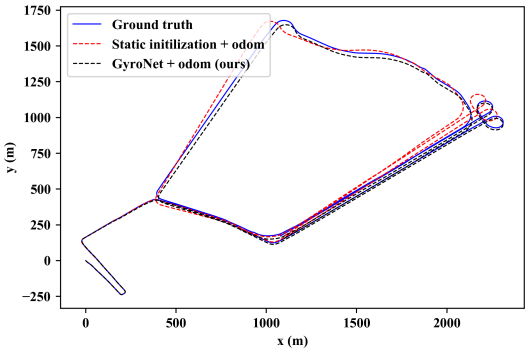


Fig. 7. The angular velocity and odometers integration result of test sequence Urban17.

measurements, while ground truth data is the yaw angle change in the time window provided by the KAIST dataset. We choose a window length of the 60s and a step size of 1s since the error is extremely small. The disadvantage is that we can not correct the gyroscope in the first time window when evaluating our network. It causes an initial error in the correction results. In RINS-W [14], they remove the results of the first time window (nearly 100s), which makes it impractical. To solve this problem, we get inspiration from the padding operation in CNN. As is shown in Fig. 8, when the current data is less than the length of the time window, we pad a time window with the existing data and transmit it to the network for prediction. It can effectively reduce the initial error.

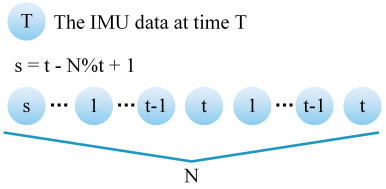


Fig. 8. The example of padding on IMU data. N is the length of the time window, and t is the current time that satisfies t < N.

2) Comparison: We compare 4 methods on KAIST Urban Dataset:

- **FoG and odometers integration:** the direct integration of wheel odometers and FoG. The wheel odometers provide linear velocity, and the FoG provides angular velocity.
- **Lwoi [2]:** learning wheel odometry and IMU errors for localization. It trains Gaussian processes on the localization error that uses IMU, FoG and wheel odometers measurements. We reproduce their results with open-source code[1].
- **RINS-W [14]:** a robust inertial navigation system on wheels. It is a state-of-the-art Kalman filter that incorporates this knowledge as pseudo-measurements for localization. It uses IMU measurements only but needs the vehicle to move 100s for initialization, which limits its practicality. We compare the results in their paper.
- **FoGNet and odometers integration (ours):** the proposed aproach, that uses IMU, FoG and wheel odometers to correct FoG measurements.

[1]https://github.com/CAOR-MINES-ParisTech/lwoi

TABLE II

ATE AND ALIGNED ATE ON KAIST URBAN06-17

Seq	FoG + odom		Lwoi [2]		**FoGNet + odom**	
	ATE (m)	Aligned ATE (m)	ATE (m)	Aligned ATE (m)	ATE (m)	Aligned ATE (m)
Urban06	126.41	38.04	56.86	22.4	**43.75**	**11.62**
Urban07	**3.02**	0.41	6.92	4.7	3.62	**0.30**
Urban08	0.87	0.38	5.87	5.14	**0.61**	**0.32**
Urban09	18.3	12.49	11.6	7.76	**2.67**	**1.34**
Urban10	50.4	14.12	54.24	9.15	**36.90**	**2.47**
Urban11	109.92	25.39	**27.79**	16.13	46.88	**14.77**
Urban12	66.93	24.21	47.34	24.66	**28.93**	**8.12**
Urban13	7.01	3.94	10.29	2.78	**1.94**	**0.51**
Urban14	35.89	13.99	49.91	10.11	**22.00**	**1.33**
Urban15	3.69	3.54	9.52	8.23	**2.59**	**1.29**
Urban16	45.21	31.37	28.43	18.95	**18.08**	**14.35**
Urban17	20.97	14.25	15.04	10.2	**7.32**	**6.91**
Average	40.72	15.22	26.98	11.68	**17.98**	**5.28**

On NCLT Dataset, we compare two methods since the open-source code of *lwoi* [2] doesn't work on it and *RINS-W* [14] doesn't test on that dataset and doesn't provide open-source code.

As the error and drift of FoG are extremely small, direct evaluation of angles is not intuitive. We compensate the raw data, calculate the trajectory of the vehicle, and consider the following two metrics:

- *Absolute Trajectory Error (ATE):* calculating the planar translation error between estimated trajectory and ground truth using *evo*[2].
- *aligned Absolute Trajectory Error (aligned ATE):* performing trajectory alignment before calculating ATE.

3) Results: We train our *FoGNet* on sequence `Urban06-13` and test on sequence `Urban14-17`. Tab. II show the ATE and aligned ATE on KAIST Urban Dataset compared with *lwoi* [2] and original odometers and FoG integration. In most cases, our approach is better than the raw data integration and *lwoi* [2].

Tab. III shows comparative results with and without padding. The effect is more evident without trajectory alignment. It will be helpful in practice since we cannot perform any alignment when driving a car. Compared with RINS-W [14], our approach performs well, as is shown in Tab. IV. By using two more sensors, we get better results and do not need a long time initialization.

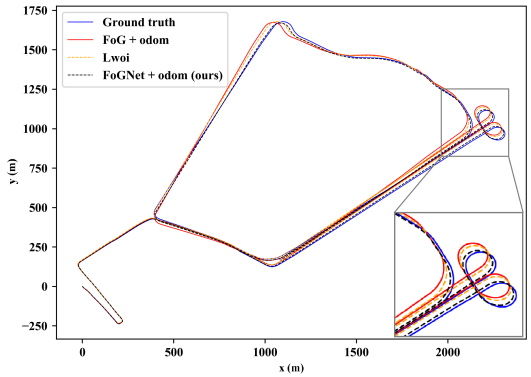


Fig. 9. The trajectory of test sequence Urban17.

[2]https://github.com/MichaelGrupp/evo

TABLE III

COMPARATIVE EXPERIMENTS ON PADDING

Method	ATE (m)	Aligned ATE (m)
No padding	19.64	5.47
Padding	**17.98**	**5.28**

TABLE IV

MEAN ATE COMPARED WITH RINS-W [14] ON TEST SETS

Seq	RINS-W [14]		**FoGNet + odom**	
	ATE (m)	Aligned ATE (m)	ATE (m)	Aligned ATE (m)
Urban15	7	5	**3**	**1**
Urban16	27	**11**	**19**	14
Urban17	13	11	**7**	**7**
Average	22	10	**9**	**7**

TABLE V

AVERAGE ATE ON NEWLY RELEASED SEQUENCE URBAN18-37

Method	ATE (m)	Aligned ATE (m)
FoG + odom	23.10	7.02
Lwoi [2]	23.79	6.02
FoGNet + odom	**14.60**	**4.31**

TABLE VI

TOTAL RUNNING TIME ON KAIST URBAN06-17

Method	Time
Lwoi [2]	1986 s
FoGNet (CPU)	1430 s
FoGNet (GPU)	220 s

To evaluate the generalization of our approach, we test the trained *lwoi* [2] and *FoGNet* models on the newly released 20 sequences of KAIST Urban Dataset. We can find out from Tab. V that our approach is still valid while the *lwoi* [2] has little improvement on aligned ATE. As we mentioned before, *lwoi* [2] trains Gaussian processes on the localization error. But the error may not meet that distribution so that our pure learning from the data approach outperforms that model.

In terms of computational efficiency, our approach is faster than *lwoi* [2]. Since *lwoi* [2] only provides the CPU version, we also test our model on the CPU. Tab. VI shows that our approach is more efficient.

On NCLT Dataset, We use 18 train sets from sequence `2012-01-08` to `2012-08-20` and the remaining nine as test sets. Tab. VII shows the ATE and aligned ATE on NCLT Dataset compared with original odometers and FoG integration. Our approach outperforms the original method.

C. Discussion

The experiments suggest that the improvement on NCLT Dataset is not as evident as KAIST Dataset. The possible reasons are as follows:

- The data acquisition platform in the NCLT Dataset is a two-wheeled Segway robot that is not as stable as the four-wheeled consumer vehicle in the KAIST Dataset.
- The wheel odometers on the Segway robot is not accurate enough, which introduces large translation error in localization. As is shown in Fig. 10, the rotation error is small, but the translation error is large in the middle of the trajectory.

TABLE VII
AVERAGE ATE ON NCLT DATASET

Method	ATE (m)	Aligned ATE (m)
FoG + odom	41.98	18.15
FoGNet + odom	**38.64**	**16.66**

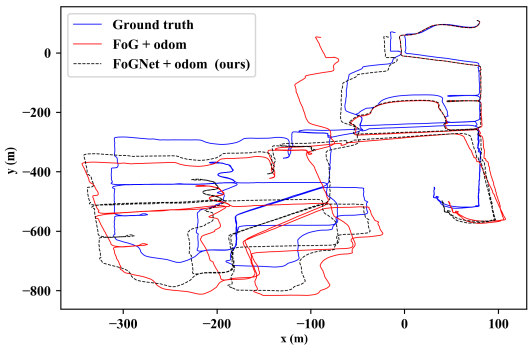


Fig. 10. The trajectory of test sequence 2012-11-17.

- The data synchronization of the NCLT Dataset is not as good as KAIST. As is shown in [16], the KVH is timestamped according to the arrival time of the sensor message. The timestamp will be confusing when data is blocked. As is shown in Fig. 11, the time difference of FoG between adjacent timestamps in the NCLT Dataset is more chaotic than that in KAIST Dataset. Other sensors have the same situation.

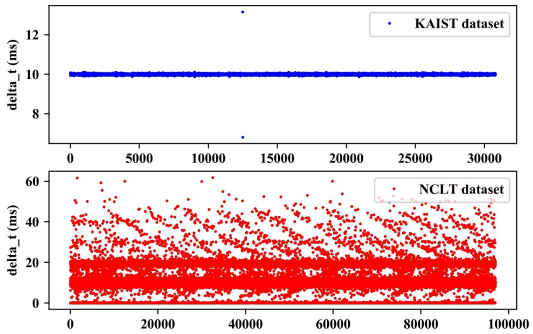


Fig. 11. The time difference of FoG between adjacent timestamps.

V. CONCLUSION

This paper proposes a novel approach for gyroscope denoising in vehicle localization through the neural network. The experiments on publicly available datasets show that our *GyroNet* has better performance on denoising than the traditional methods, our *FoGNet* outperforms the state-of-the-art method *lwoi* [2] and is more efficient.

In future work, we plan to compensate for the drift of the wheel odometers and the gyroscope at the same time to get more accurate results on NCLT Dataset. Furthermore, we will combine the *GyroNet* and the *FoGNet* together, which takes the denoising result of *GyroNet* as a pseudo-FoG measurement and replace the real FoG measurements in *FoGNet*. Thus, the costly FoG is only used for data collection in the training process, and we can get an accurate location based on low-cost IMU and wheel odometers. Besides, this work is an attempt to utilize machine learning in the sensor data processing. It approves that the model can learn some non-artificial parameters from sequence data. In application, the easiest way is to use the network to denoise raw sensor data in the visual-inertial navigation system. We will focus on learning uncertainty to distinguish sensor failure and get sensor confidence for multi-sensor fusion algorithm.

VI. ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China under Grant 61836015 and the Key Research and Development Program of Guangdong Province of China under Grant 2019B010120001.

REFERENCES

[1] C. Chen, C. X. Lu, A. Markham, and N. Trigoni, "Ionet: Learning to cure the curse of drift in inertial odometry," in *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

[2] M. BROSSARD and S. BONNABEL, "Learning wheel odometry and imu errors for localization," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 291–297, May 2019.

[3] I. Skog, P. Handel, J. Nilsson, and J. Rantakokko, "Zero-velocity detectionan algorithm evaluation," *IEEE Transactions on Biomedical Engineering*, vol. 57, pp. 2657–2666, Nov 2010.

[4] I. Skog, J. Nilsson, and P. Hndel, "Evaluation of zero-velocity detectors for foot-mounted inertial navigation systems," in *2010 International Conference on Indoor Positioning and Indoor Navigation*, pp. 1–6, Sep. 2010.

[5] A. Olivares, J. Ramirez, J. M. Górriz, G. Olivares, and M. Damas, "Detection of (in) activity periods in human body motion using inertial sensors: a comparative study," *Sensors*, vol. 12, no. 5, pp. 5791–5814, 2012.

[6] E. Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," *IEEE Computer Graphics and Applications*, vol. 25, pp. 38–46, Nov 2005.

[7] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Robotics and automation, 2007 IEEE international conference on*, pp. 3565–3572, IEEE, 2007.

[8] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[9] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[10] K. J. Wu, C. X. Guo, G. Georgiou, and S. I. Roumeliotis, "Vins on wheels," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5155–5162, May 2017.

[11] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem," *CoRR*, vol. abs/1701.08376, 2017.

[12] B. Wagstaff and J. Kelly, "Lstm-based zero-velocity detection for robust inertial navigation," in *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8, Sep. 2018.

[13] L. Han, Y. Lin, G. Du, and S. Lian, "Deepvio: Self-supervised deep learning of monocular visual inertial odometry using 3d geometric constraints," *CoRR*, vol. abs/1906.11435, 2019.

[14] M. Brossard, A. Barrau, and S. Bonnabel, "Rins-w: Robust inertial navigation system on wheels," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2068–2075, Nov 2019.

[15] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *The International Journal of Robotics Research*, p. 0278364919843996, 2019.

[16] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of Michigan North Campus long-term vision and lidar dataset," *International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2015.