RGNet: Robust Gravity Estimation Neural Network for IMU-based Localization Using Smartphone

Takuto Yoshida[1,2], Kenta Urano[1], Shunsuke Aoki[3], Takuro Yonezawa[1], and Nobuo Kawaguchi[1]
[1]*Graduate School of Enginnering, Nagoya University*
[2]*E-mail: takuto@ucl.nuee.nagoya-u.ac.jp*
[3]*National Institute of Informatics*

***Abstract*—With the rapid development of Micro Electro-Mechanical Systems (MEMS) technologies, indoor navigation and localization with Inertial Measurement Unit (IMU) has been increasingly feasible. IMU-based indoor localization is a low-cost, energy-efficient, and infrastructure-free approach. There are various methods for it, and most of them require gravity estimation (e.g. the projection of angular velocity, the extraction of horizontal acceleration). In particular, when you use a smartphone, it changes the orientation of its IMU frequently, therefore the gravity estimation needs to be more robust to sensor orientation and its noise. In this paper, we propose a gravity estimation method based on deep learning called RGNet (Robust Gravity Estimation Neural Network) that is robust to sensor orientation and noise. We train an LSTM (Long short-term memory)-based neural network that estimates gravity from acceleration and angular velocity. Furthermore, for the problem that it is difficult to prepare the ground truth of gravity directly, we propose a method to train the gravity estimation neural network indirectly using the heading, taking advantage of the fact that the heading can be estimated from the gravity and angular velocity. The evaluation results show that the accuracy of the proposed method outperformed the baseline gravity estimation method (Android API, Low-pass filter, and Extended Kalman filter). We also confirmed that the accuracy of IMU-based localization is affected by the difference in gravity estimation methods.**

***Index Terms*—Dead reckoning, Indoor navigation, Recurrent neural networks, Smartphone sensors**

I. Introduction

A variety of location-based applications and services require indoor navigation and localization technologies. For example, map navigation and human-mobility analysis are heavily relaying on indoor navigation and localization. There are two types of technologies: 1) infrastructure-based method; 2) Inertial measurement unit (IMU)-based method. First, the infrastructure-based method includes Bluetooth low energy (BLE) beacon, ultra-wideband (UWB), wireless fidelity (Wi-Fi), and radio-frequency identification (RFID) [1]–[5]. It can achieve accurate and stable position estimation if you can deploy a large number of signal transmitting devices. However, they cost a lot, and most of it requires adjusting the parameters depending on the indoor situation. On the other hand, the IMU-based method is an approach that uses IMU data to continuously calculate by dead reckoning the position. Therefore, the method becomes low-cost, energy-efficient, and infrastructure-free one. Furthermore, the recent advance of Micro Electro-Mechanical Systems (MEMS) technology enables us to apply it to small devices such as smartphones and wearable devices. In fact, many researchers have study of IMU-based indoor navigation and localization [6]–[11].

A low-cost IMU of the smartphone may be noisy and bias, which can cause estimation errors. Therefore, various methods have tried to solve this problem. Kang et al. proposed SmartPDR [12], which integrates angular velocity and geomagnetism and complement each other's shortcomings. PCA (Principal Component Analysis)-based positioning methods [13]–[15] have also been proposed, including RMPCA [16] proposed by Deng et al. This method provides the heading estimation that is robust to changes in sensor orientation and sensor noise. These methods all use 3-axis gravity $\boldsymbol{g}$: SmartPDR uses it to extract z-axis angular velocity ω_z from 3-axis angular velocity $\boldsymbol{\omega}$ from Equation (1); PCA-based method uses gravity to extract horizontal acceleration $\boldsymbol{a}_h$ from 3-axis acceleration $\boldsymbol{a}$ form Equation (2).

$$\omega_z = \frac{\boldsymbol{\omega} \cdot \boldsymbol{g}}{|\boldsymbol{g}|} \tag{1}$$

$$\boldsymbol{a}_h = \boldsymbol{a} - \left(\frac{\boldsymbol{a} \cdot \boldsymbol{g}}{\boldsymbol{g} \cdot \boldsymbol{g}}\right)\boldsymbol{g} \tag{2}$$

The IMU can measure the raw acceleration, but not the gravity directly. This means that gravity estimation plays a significant role in IMU-based indoor localization. In addition, if you use smartphone sensors, the sensor orientation changes frequently, so you need a gravity estimation method that is robust to sensor orientation and noise. However, there are only a few studies discussing methods for estimating gravity [17], [18].

Therefore, we propose a method for estimating the gravity using deep learning that is robust to sensor orientation and sensor noise. Specifically, we train an LSTM-based neural network called RGNet (Robust Gravity Estimation Neural Network) to estimate 3-axis gravity from 3-axis acceleration and 3-axis angular velocity using a large amount of data. The biggest problem with this method is that it is difficult to prepare high-precision gravity data that can be used as the ground truth when we train RGNet. However, we can solve this problem by indirectly training method. As in general supervised learning, it calculates the gradient from the loss of estimates and updates the parameters using it. However, the loss is calculated from the estimated heading, not estimated gravity. We estimate the heading by integrating the z-axis angular velocity projected onto 3D vector estimated by a neural network. As a result, the neural network learns how to estimate the gravity indirectly. In this paper, we also examine the effect of the accuracy of the gravity on the accuracy of localization using Oxford Inertial Odometry Dataset (OxIOD), three baseline methods (Android API, Low-pass filter, and Extended Kalman filter). The evaluation results show that the accuracy of the proposed method outperformed the baseline method. We also confirmed that the accuracy of localization is affected by the different gravity estimation methods.

II. Related work

There are various methods for IMU-based indoor localization. Kang et al. have proposed a smartphone-based pedestrian dead reckoning system (PDR) called SmartPDR [12]. It integrates angular velocity and geomagnetism to complement each other's shortcomings. Deng et al. have proposed a PCA-based localization system called RMPCA [16]. They combine rotation matrix and PCA to achieve heading estimation regardless of sensor orientation and position. There are various other PCA-based methods have also been proposed [13]–[15]. All of these methods commonly use gravity. The SmartPDR uses it to extract the z-axis angular velocity from the 3-axis angular velocity from Equation (1). PCA-based method uses it to extract the horizontal acceleration from Equation (2). IMU can measure raw acceleration, but cannot directly measure gravity. This means that gravity estimation is required in many IMU-based indoor localization.

There are two main methods for gravity estimation for the IMU-based indoor localization method, Low-pass filter (LPF) [17] and Extended Kalman filter (EKF) method [18]. The accelerometer measures raw acceleration consisting of linear acceleration and gravity. LPF can remove the linear acceleration from raw acceleration and extract gravity since the linear acceleration has a higher frequency component than gravity. It is often used in the IMU-based method because it is simple and clear. However, there is a latency between the gravity estimated by it and the ground truth. Referring to robustness to noise, if the gravity contains a high-frequency component, the LPF will mistakenly judge it to be linear acceleration. On the other hand, if the acceleration contains low-frequency noise, the LPF will mistakenly judge it to be gravity. EKF method [17] uses angular velocity, initial gravity, and observed gravity. The angular velocity is sensitive to changes in the axis of the sensor. Therefore, it is useful information for tracking changes in the gravity direction. EKF method estimates the gravity by fusing the observed gravity with the gravity updated by the angular velocity from the initial gravity. It can track the dynamics change of the sensor frame. However, it has a problem that we need to prepare another method for estimating initial gravity and observed gravity. Manos et al. [18] have used the gravity which LPF estimates as observed gravity. However, we suspect that it will lead to the LPF problem again. Therefore, we require a new high precision and robust gravity estimation method and propose a method for estimating the gravity using deep learning.

Training a deep neural network requires a large number of data. A dataset that includes inertial sensor data and ground-truth motion trajectories for IMU-based indoor localization has been released recently. Robust IMU double integration (RIDI) dataset [19] has inertial sensor data and 3D motion trajectory collected by Google Tango. Google Tango provides the ground truth of device trajectory and orientation using visual-intertial odometry. In order to use Google Tango, you need a special device equipped with a fisheye lens and deep perception sensor. Oxford Inertial Odometry Dataset (OxIOD) [20] has an inertial sensor data containing enough types of users, device position, and gait to test the robustness of the indoor localization system. It also provides high precision 3D trajectory data collected by an optical motion capturing system (Vicon) as an advantage in supervised learning. Robust Neural Inertial Navigation (RoNIN) dataset [21] is large inertial navigation dataset. It has 42.7 hours of IMU-motion data and various modalities data (e.g. a bag, placing deep inside a pocket, picking up by hand, while walking, sitting, or wandering around) collected from 100 human subjects. These datasets enable a data-driven IMU-based indoor localization. In this paper, we use OxIOD to train our proposed model.

III. RGNet

RGNet is a new data-driven method for robust gravity estimation. It estimates gravity from acceleration and angular velocity using a model based on deep neural network. In this section, we describe the three key points of RGNet. We first explain the architecture of RGNet which consists of three nodes, gravity estimation node, projection node, and integration node. Gravity estimation node is a neural network based on LSTM to estimate gravity. Projection node calculates the horizontal angular velocity from the estimated gravity and angular velocity. Integration node calculates the heading by integrating the horizontal angular velocity. The second key point is a truncated BPTT based training method which allows for stable training of long-time inertial motion sensor data by dividing it into short segments. Finally, we explain a loss function consisting of two terms: squared error of heading and mean squared error (MSE) of acceleration and gravity as a regularization term. It allows the learning of the model to converge optimally.

A. Definition of coordinate frames and symbols

In order to explain our proposed method, we introduce a number of coordinate systems and symbols. Figure 1 shows two coordinate systems and the symbols of the physical quantities in those coordinate systems when a pedestrian holds a smartphone. Device coordinate system (DCS) is the coordinate frame of the moving IMU embedded in a smartphone. Global coodinate system (GCS) is the reference coordinate for indoor localization, and the z-axis is aligned with the negative gravity direction. $\boldsymbol{a}^{DCS}$ and $\boldsymbol{\omega}^{DCS}$ are acceleration and angular velocity collected by a smartphone. $\boldsymbol{\omega}^{DCS}$ has already had the

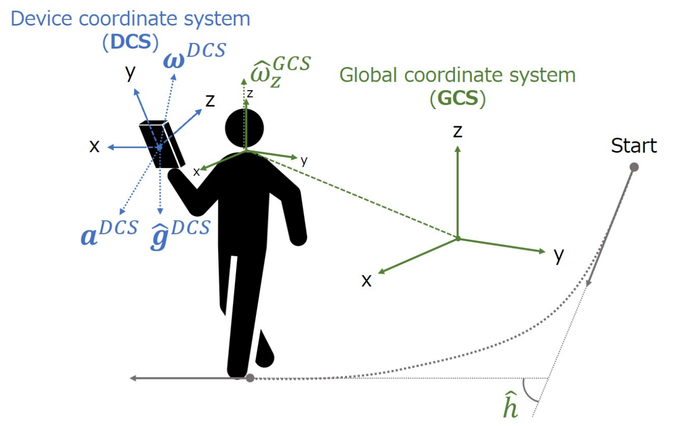


Fig. 1: Two coordinate systems and the symbols of the physical quantities - pedestrian holds a smartphone.

offset removed. $\hat{\boldsymbol{g}}^{DCS}$ is the gravity that our proposed method estimates. $\hat{\omega}_z^{GCS}$ is the horizontal angular velocity which is projection of $\boldsymbol{\omega}^{DCS}$ to $\hat{\boldsymbol{g}}^{DCS}$. $\hat{h}$ is a heading obtained by integrating $\hat{\omega}_z^{GCS}$.

B. RGNet architecture

Figure 2 shows the overall RGNet architecture. It consists of three main nodes.

- **Gravity estimation node** estimates $\hat{\boldsymbol{g}}^{DCS}$ from $\boldsymbol{a}^{DCS}$ and $\boldsymbol{\omega}^{DCS}$ using a neural network based on LSTM.
- **Projection node** calculates $\hat{\omega}_z^{GCS}$ by projecting $\boldsymbol{\omega}^{DCS}$ to $\hat{\boldsymbol{g}}^{DCS}$.
- **Integration node** calculates heading by integrating time-series $\hat{\omega}_z^{GCS}$.

The first gravity estimation node is a neural network based on LSTM that estimates $\hat{\boldsymbol{g}}^{DCS}$. Input data is a 6-dimensional features, concatenated with $\boldsymbol{a}^{DCS}$ and $\boldsymbol{\omega}^{DCS}$. The LSTM learns the features of it and converts them into different dimensions features, called the hidden state and cell state. The hidden state is propagated to the fully connected layer. It converts the hidden state to $\hat{\boldsymbol{g}}^{DCS}$. Its parameters at each node are common. Furthermore, the hidden state and cell state of LSTM are propagated to the next LSTM nodes. It enables RGNet to accept the arbitrary length of time series inertial sensor data.

The second is a projection node that calculates horizontal angular velocity $\hat{\omega}_z^{GCS}$ from $\boldsymbol{\omega}^{DCS}$ and $\hat{\boldsymbol{g}}^{DCS}$. Its process is as follows

$$\hat{\omega}_z^{GCS} = -\frac{\boldsymbol{\omega}^{DCS} \cdot \hat{\boldsymbol{g}}^{DCS}}{\|\hat{\boldsymbol{g}}^{DCS}\|} \tag{3}$$

Equation (3) projects $\boldsymbol{\omega}^{DCS}$ to $\hat{\boldsymbol{g}}^{DCS}$ to obtain $\hat{\omega}_z^{GCS}$. We assume that the axis of pedestrian's heading change aligne with gravity. Therefore, $\hat{\omega}_z^{GCS}$ means the heading change per unit of time.

The third is an integration node that integrates $\hat{\omega}_z^{GCS}$ to obtain heading $\hat{h}$.

$$\hat{h} = \sum \hat{\omega}_z^{GCS} dt \tag{4}$$

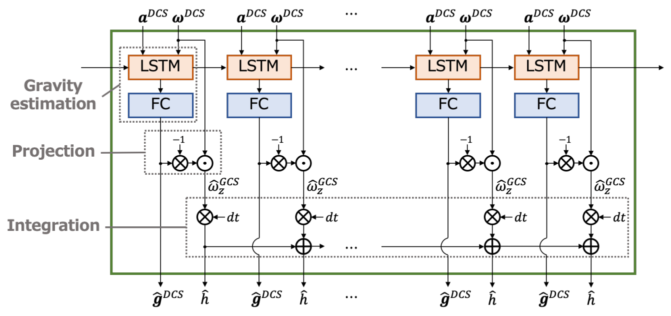


Fig. 2: The architecture of RGNet.

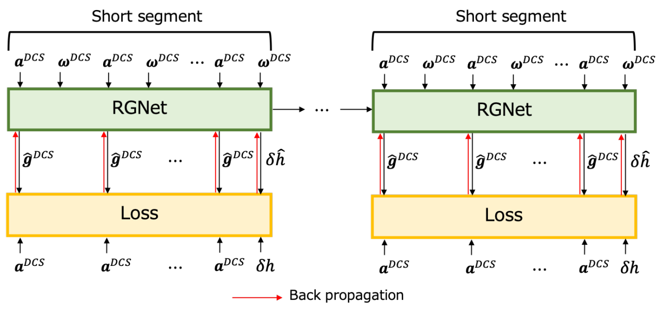


Fig. 3: Training method for RGNet using truncated BPTT.

The dt indicates the sampling interval. Larger sampling intervals result in larger integration errors because our method uses a simple numerical integration. Therefore, we set it short sampling interval, 0.01 sec.

C. Training processes

RGNet can handle arbitrary time-series length sensor data in the estimation phase. However, in the training phase, we have to be careful with time-series data length especially when it is long-term. The concerns with training RGNet on long term data are as following:

- A lot of memory resources consumed by LSTM (e.g. automatic differentiation and gradient information)
- A mis-propagation of gradient information through long-term (vanishing/exploding gradient problem)

Therefore, we adapt a truncated BPTT [22], [23] to train RGNet to avoid these concerns. Truncated BPTT is a method for learning time series models in which the forward propagation is allowed to propagate without interruption, while the backward propagation connections are truncated to an appropriate length. Figure 3 shows the training phase of RGNet which is divided into short segments. It maintains the forward propagation link (black arrows), but the backward link (red arrows) is truncated between short segments. A good choice of segment length contributes to the success of model training.

D. Gravity and heading loss

The design of the loss function is very important in the maximum likelihood estimation method by minimizing the loss. In our proposed method, it corresponds to the training of a neural network for gravity estimation. However, most datasets do not have the ground truth of gravity. Therefore, we provide a loss function for RGNet to learn a model to regress gravity indirectly.

RGNet estimates gravity $\hat{\boldsymbol{g}}^{DCS}$ and the heading change $\delta\hat{h}$ between short segments as shown in Figure 3. The hidden layer of the LSTM is propagated between each short segment, but $\hat{h}$ is not. Therefore, the initial $\hat{h}$ for each short segment is 0, and $\delta\hat{h}$ represents the change heading within the short segment. We calculate the back propagation loss from $\hat{\boldsymbol{g}}^{DCS}$, $\delta\hat{h}$ and $\boldsymbol{a}^{DCS}$, δh each short segments. The δh in the short segment is defined by the difference in the ground truth heading in the GCS between the end point and the start point in the short segment. Our proposed loss function consists of 2 terms: an heading loss term and a gravity regularization term as

$$\mathcal{L} = \|\delta\hat{h} - \delta h\|_2^2 + \kappa \frac{\sum_i^N \|\hat{\boldsymbol{g}}^{DCS} - \boldsymbol{a}^{DCS}\|_2^2}{N} \tag{5}$$

where κ is a coefficient for adjusting the weight of the second term. N is segment length, in other words, the number of samples in the short segment. The first term in equation (5), an heading loss term, calculates the squared error between $\delta\hat{h}$ and δh. The loss is propagated as a gradient to each neural network through the integration nodes and the projection nodes. The second term in equation (5), a gravity regularization term calculates the mean square error (MSE) between time-series gravity and acceleration. It takes advantage of the similarity between the low-frequency features of acceleration and gravity, which works to support learning convergence. The loss is propagated as a gradient to each gravity estimation neural network.

IV. Train model

In this section, we discuss the training method for RGNet. The goal of this section is to explain training and evaluation method and to find the optimal hyperparameters for training and neural network.

We implement the RGNet architecture using Pytorch [24] and use the GPU of NVIDIA RTX 2080 Ti for training. We split OxIOD into training, validation, hyperparameters testing, and testing. We use training and validation data to train our models. The hyperparameters testing data is used to grid-search over a parameter grid of training and neural network. The test data is used to evaluate the method and to compare the proposed method with the conventional methods. We use Adam [25] for optimization. Figure 4 shows the example of train and validation loss result. They find that the loss decreases and converges as the epoch progresses. We choose the best model where the validation loss is minimal. In this example, it is 500 epochs.

We evaluate RGNet in two ways: heading and trajectory estimation. We use the Relative Angle Error (RAE) to evaluate heading, defined as the average of the Root Mean Squared Error (RMSE) between of orientation [deg] per 1 minute. This metric evaluates the accuracy of orientation estimation fairly, regardless of route length or walking time, because it calculates a minute-by-minute error. We use the heading estimated by RGNet for our evaluation. We use the Absolute Trajectory Error (ATE) to evaluate trajectory, defined as the RMSE between estimated and ground truth trajectories [26]. This is a standard metrics that are also used in visual odometry systems. We calculate the trajectory using the heading estimated by RGNet and the speed calculated from the ground truth position.

A. Hyperparameters for training

The hyperparameters for traing are batch size, learning rate, N, and κ. Table I shows training hyperparameters which are determined by grid-search.

Batch size and learning rate have a significant impact on model generalization performance [27]. The appropriate values of them depend on the amount of the training dataset. If the batch size is too small or too large, it reduces the generalization performance [28]. N is the segment length to be divided in truncated BPTT. It is an important value when backpropagation because it determines the extent to which historical information is used to calculate the gradient. If it is too short, RGNet cannot learn enough time-series information. On the other hand, if it is too long, it will consume a lot of computer resources and cause a vanishing/exploding gradient problem. We experiment to evaluate the impact of the segment length N of the truncated BPTT using the hyperparameters testing data of OxIOD. Table III shows the evaluation result for each N of 200, 400, 800, 1600. The best estimation accuracy for orientation and trajectory is obtained for the model when N is 800. κ is a coefficient that is the weight of the regulation term in equation (5). It takes advantage of the similarity between the low-frequency features of acceleration and gravity, which works to support learning convergence. We experiment to evaluate the impact of κ using the hyperparameters testing data of OxIOD. Table IV shows the evaluation result for each κ of 0, 0.01, 0.1, 1, 5. The best estimation accuracy for orientation and trajectory is obtained for the model when κ is 0.1.

B. Hyperparameters for neural network

Table II shows hyperparameters of the neural network that estimates gravity. A num layers is the number of recurrent layers of LSTM. A hidden size is the number of features in the hidden state of LSTM. It also matches the input feature size of a fully connected layer. A dropout is a probability of an input tensor element to be zeroed. It has the effect of preventing over-fitting. We have determined these parameters by grid-search.

V. Evaluation

In this section, we evaluate the robustness of RGNet in four perspectives: gait, device position, large-scale floor trajectory, and dataset modalities. We have three comparative methods: Android API, LPF, and EKF. The Android API refers to the motion sensor API provided by Android 10. The LPF can

TABLE I: Hyperparameters for training.

Parameters	Value
batch size	512
learning rate	0.001
N	800
κ	0.1

TABLE II: Hyperparameters for our neural network.

Parameters	Value
num layers	1
hidden size	64
dropout	0.25

extract the low-frequency component as gravity from acceleration [18]. EKF estimates gravity using the initial gravity and angular velocity and updates it using the observed gravity [17]. In this paper, we use the gravity estimated by the LPF as the initial gravity and the observed gravity.

A. *Overall result*

Table V is the evaluation result of the overall of OxIOD. Our proposed method outperforms conventional methods. It succeeds in improving accuracy by about 30-40%. Figure 5 shows an example of a gravity estimation result. Our proposed method is able to estimate higher frequency component than other methods. Figure 6 shows the orientation estimated by the gravity in Figure 5. Our proposed method follows ground truth exactly, whereas the other methods are affected by the cumulative error. This result indicates that gravity estimation is a key factor in orientation estimation.

B. *Robustness to gait*

We evaluate the robustness to gait using walking, slow-walking, and running data of OxIOD testing. Table VI shows the evaluation result. Our proposed method has the highest accuracy in the walking and slow-walking evaluation result. In walking, the RAE of the proposed method is 23.11 [deg]. In slow-walking, the RAE of the proposed method is 7.66 [deg].

However, in running results, the API has the highest accuracy, and our proposed method is the third. This result can be attributed to the amount of training data. Table VII shows the amount of training data per gait. According to Table VII, the amount of running data is the smallest of the three categories. From this fact, it shows the amount of training data affects the estimation accuracy of the model for each gait. This indicates that the robustness of the proposed method to gait can be improved by increasing the amount of training data.

C. *Robustness to device position*

We evaluate the robustness of the device position using handheld, handbag, and pocket data of OxIOD testing. Table VIII shows device position evaluation result. It is found that the accuracy of the proposed method is the highest in all device positions. In handheld, the RAE of the proposed method is

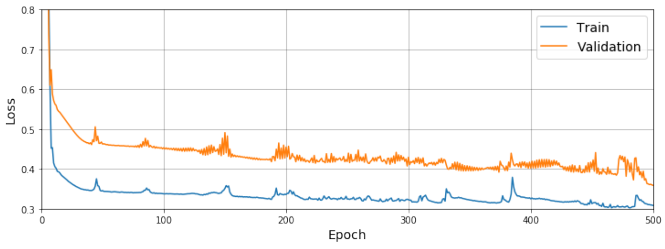


Fig. 4: Train and validation loss for training processes of RGNet.

TABLE III: Evaluation results for each segment length.

Metrics	N				
	200	400	800	1600	3200
RAE [deg]	41.16	18.15	**13.38**	13.45	25.15
ATE [m]	13.03	6.09	**3.96**	4.87	9.52

34.70 [deg]. In the handbag, the RAE of the proposed method is 31.14 [deg]. In the pocket, the RAE of the proposed method is 10.40 [deg]. The 67.45 [deg] error of LPF is much larger than the others. We suspect that the high-frequency changes of gravity direction caused by the shaking in the pockets confused the LPF. It shows our proposed method works robustly even in a swaying position that confuses the LPF, such as a pocket.

D. *Robustness to large-scale floor trajectory*

Most of the OxIOD is trajectory data in a narrow area collected in the Vicon room. However, it also contains large-scale floor data using Google Tango for evaluation of wide-area trajectory. Table IX shows the results of the evaluation of the large-scale floor data, which is an environment that is assumed to be used for user navigation and human-flow analysis in shopping malls and underground malls. The results of RAE and ATE show that the proposed method has the highest estimation accuracy for orientation and trajectory. The RAE of the proposed method is 13.48 [deg]. the ATE of the proposed method is 3.96 [m].

Figure 7 shows an example of orientation and trajectory, which shows that the estimation of orientation has a significant impact on the accuracy of trajectory estimation. In Figure 7a, the API, LPF and EKF fail to close the loop of estimated trajectory since the estimation orientation is more than 90 [deg] from the ground truth at 140 [sec]. On the other hand, the proposed method has almost no error in orientation estimation at 140 [sec], and succeed in closing the trajectory loop. In Figure 7b, while the API and EKF have errors in orientation estimation that accumulate over time, the LPF and RGNet have almost no accumulated errors.

E. *Robustness to other dataset*

What is important in the data-driven method is that the models you create work well with a variety of data. Therefore, we evaluate the RGNet model trained by OxIOD on a different dataset, which is Nagoya University Inertial Odometry Dataset (NUIOD). NUIOD is a dataset for IMU-based indoor localization that we collected using Google Tango and smartphone

TABLE IV: Evaluation results for each κ.

Metrics	κ				
	0	0.01	0.1	1	5
RAE [deg]	18.87	21.56	**13.38**	54.89	45.54
ATE [m]	5.62	8.23	**3.96**	15.35	14.73

TABLE V: Evaluation results using OxIOD.

Metrics	API	LPF	EKF	RGNet
RAE [deg]	22.44	28.20	20.67	**18.66**
ATE [m]	6.13	4.53	4.89	**3.63**

IMU sensors in as shown Figure 8. We use data included walking, fast-walking, and staying, which is a total of 10562 m.

Table X shows the evaluation results using NUIOD. The API is not included in the evaluation because NUIOD does not have a gravity data of API. Our proposed method outperforms the conventional method in the orientation and trajectory estimation. This result shows that RGNet's model trained by OxIOD can be adapted to estimate other datasets. The accuracy of the proposed method has only a slight advantage over conventional methods. Further improvement of the model is the next step.

VI. Conclusion

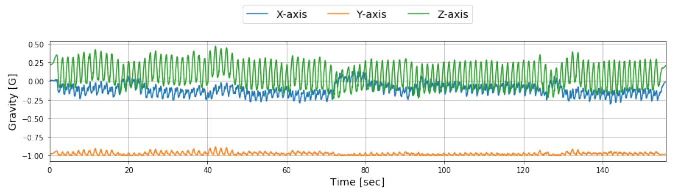
In this paper, we focused on the problem of gravity and orientation estimation threatened by sensor orientation change and its noise. To solve this problem, we proposed a new data-driven method to estimate gravity robustly, which we named RGNet. It estimates gravity from acceleration and angular velocity with using a neural network based on LSTM, which is consists of three nodes, gravity estimation node, projection node, and integration node. We also proposed a method to train the gravity estimation neural network indirectly using the heading, taking advantage of the fact that the heading can be estimated from the gravity and angular velocity for the problem that it is difficult to prepare the ground truth of gravity directly. In addition, we adapted a truncated BPTT and an original loss function as techniques for this training method to work well. Finally, we conducted the experiments to prove for RGNet to improve the accuracy and robustness through comparison with conventional methods. As a result, we confirmed that our proposed method is more accurate and robust than conventional methods. In future work, we plan to improve the performance of RGNet by training it on more dataset.

Acknowledgment

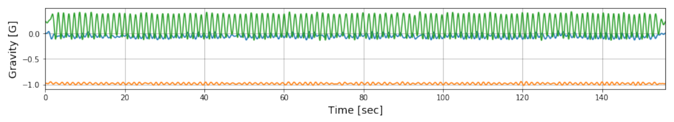
This work is supported by JSPS KAKENHI Grant Number JP17H01762.

References

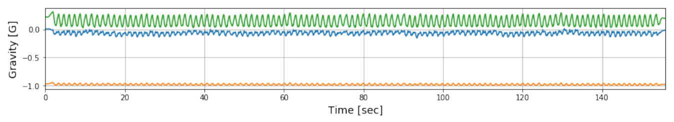
[1] M. Ji, J. Kim, J. Jeon, and Y. Cho. Analysis of positioning accuracy corresponding to the number of ble beacons in indoor positioning system. In *2015 17th International Conference on Advanced Communication Technology (ICACT)*, pages 92–95, 2015.



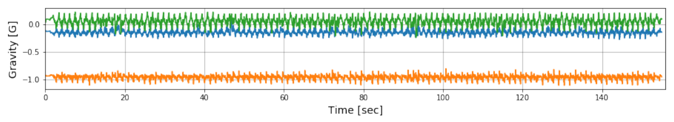
(a) API



(b) LPF



(c) EKF



(d) RGNet

Fig. 5: Estimated gravity vectors by our approach and baseline approach.

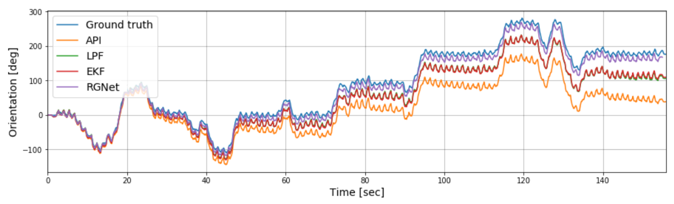


Fig. 6: Estimated orientations by our approach and baseline approach.

[2] T. Gigl, G. J. M. Janssen, V. Dizdarevic, K. Witrisal, and Z. Irahhauten. Analysis of a uwb indoor positioning system based on received signal strength. In *2007 4th Workshop on Positioning, Navigation and Communication*, pages 97–101, 2007.

[3] A. Bekkali, H. Sanson, and M. Matsumoto. Rfid indoor positioning based on probabilistic rfid map and kalman filtering. In *Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007)*, pages 21–21, 2007.

[4] Chenshu Wu, Jingao Xu, Zheng Yang, Nicholas D. Lane, and Zuwei Yin. Gain without pain: Accurate wifi-based localization using fingerprint spatial gradient. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(2), June 2017.

[5] Wei Gong and Jiangchuan Liu. Sifi: Pushing the limit of time-based wifi localization using a single commodity access point. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(1), March 2018.

[6] Raghav H. Venkatnarayan and Muhammad Shahzad. Enhancing indoor inertial odometry with wifi. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(2), June 2019.

[7] Zhao Tian, Yu-Lin Wei, Wei-Nin Chang, Xi Xiong, Changxi Zheng, Hsin-Mu Tsai, Kate Ching-Ju Lin, and Xia Zhou. Augmenting indoor inertial tracking with polarized light. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '18, page 362–375, New York, NY, USA, 2018. Association

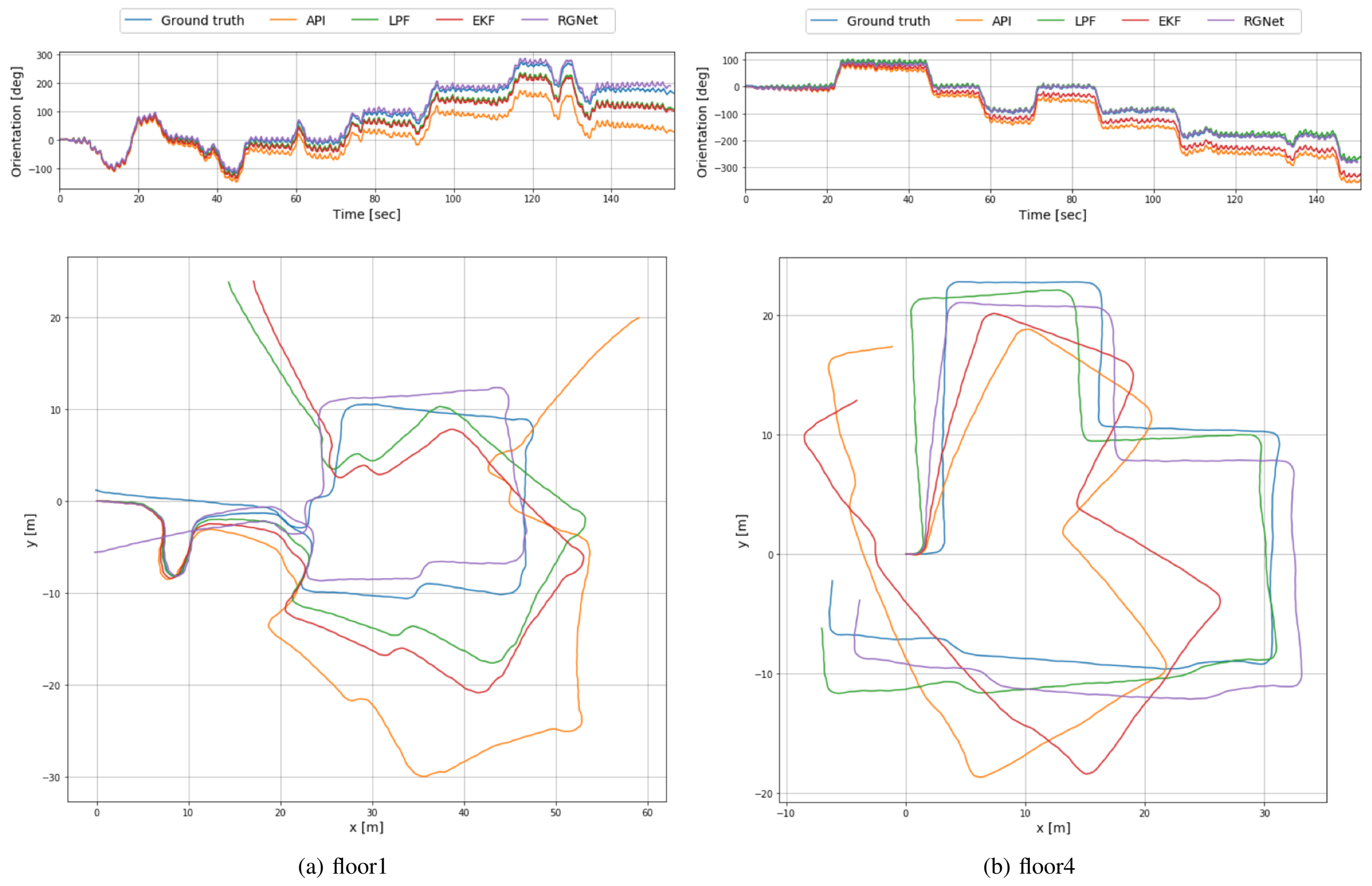


Fig. 7: Estimated orientations and trajectories for large-scale floor.

TABLE VI: Evaluation results for each gait: walking, slow-walking, and running.

Metrics	Gait	API	LPF	EKF	RGNet
RAE [deg]	Walking	33.38	34.06	32.05	**23.11**
	Slow-walking	8.57	8.82	8.77	**7.66**
	Running	**11.57**	26.97	12.64	18.05

TABLE VII: Breakdown time for each gait in training data.

	Walking	Slow-walking	Runing
Time [sec]	8681	4251	3731

TABLE VIII: Evaluation results for each device position: handheld, handbag, and pocket.

Metrics	device position	API	LPF	EKF	RGNet
RAE [deg]	Handheld	83.65	88.67	82.57	**34.70**
	Handbag	39.58	42.38	39.55	**31.14**
	Pocket	12.89	67.45	16.41	**10.40**

TABLE IX: Evaluation results for large-scale floor.

Metrics	API	LPF	EKF	RGNet
RAE [deg]	22.69	17.66	18.72	**13.48**
ATE [m]	7.07	4.91	5.53	**3.96**

for Computing Machinery.

[8] Zheng Yang, Chenshu Wu, Zimu Zhou, Xinglin Zhang, Xu Wang, and Yunhao Liu. Mobility increases localizability: A survey on wireless indoor localization using inertial sensors. *ACM Comput. Surv.*, 47(3), April 2015.

[9] Katsuhiko Kaji, Masaaki Abe, Weimin Wang, Kei Hiroi, and Nobuo Kawaguchi. Ubicomp/iswc 2015 pdr challenge corpus. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, UbiComp '16, page 696–704, New York, NY, USA, 2016. Association for Computing Machinery.

[10] Shun Yoshimi, Kohei Kanagu, Masahiro Mochizuki, Kazuya Murao, and Nobuhiko Nishio. Pdr trajectory estimation using pedestrian-space constraints: Real world evaluations. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, UbiComp/ISWC'15 Adjunct, page 1499–1508, New York, NY, USA, 2015. Association for Computing Machinery.

[11] Koki Tamura, Hiroto Asai, and Nobuhiko Nishio. Pdr with head swing detection only using hearable device. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, UbiComp/ISWC '19 Adjunct, page 843–848, New York, NY, USA, 2019. Association for Computing Machinery.

[12] Wonho Kang and Youngnam Han. Smartpdr: Smartphone-based pedestrian dead reckoning for indoor localization. *IEEE Sensors Journal*, 15:1–1, 01 2014.

[13] Kai Kunze, Paul Lukowicz, Kurt Partridge, and Bo Begole. Which way am i facing: Inferring horizontal device orientation from an accelerometer signal. In *2009 International Symposium on Wearable Computers*,

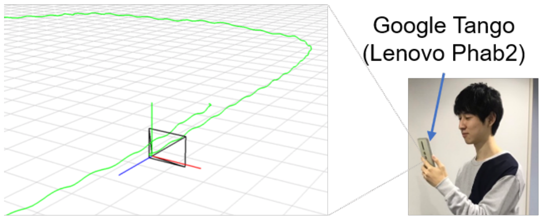


Fig. 8: Data collection with Google Tango.

TABLE X: Evaluation result with NUIOD.

Metrics	LPF	EKF	RGNet
RAE [deg]	43.00	43.87	**42.23**
ATE [m]	1.42	1.46	**1.36**

pages 149–150, 2009.

[14] Kai Kunze, Paul Lukowicz, Kurt Partridge, and Bo Begole. Which way am i facing: Inferring horizontal device orientation from an accelerometer signal. In *2009 International Symposium on Wearable Computers*, pages 149–150, 2009.

[15] Ryoji Ban, Katsuhiko Kaji, Kei Hiroi, and Nobuo Kawaguchi. Indoor positioning method integrating pedestrian dead reckoning with magnetic field and wifi fingerprints. In *2015 Eighth International Conference on Mobile Computing and Ubiquitous Networking (ICMU)*, pages 167–172, 2015.

[16] Zhi-An Deng, Guofeng Wang, Ying Hu, and Di Wu. Heading estimation for indoor pedestrian navigation using a smartphone in the pocket. *Sensors*, 15(9):21518–21536, 2015.

[17] A. Manos, I. Klein, and T. Hazan. Gravity direction estimation and heading determination for pedestrian navigation. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 206–212, 2018.

[18] Adi Manos, Itzik Klein, and T. Hazan. Gravity-based methods for heading computation in pedestrian dead reckoning. *Sensors (Basel, Switzerland)*, 19, 2019.

[19] Hang Yan, Qi Shan, and Yasutaka Furukawa. RIDI: robust IMU double integration. *CoRR*, abs/1712.09004, 2017.

[20] Changhao Chen, Peijun Zhao, Chris Xiaoxuan Lu, Wei Wang, Andrew Markham, and Niki Trigoni. Oxiod: The dataset for deep inertial odometry. *CoRR*, abs/1809.07491, 2018.

[21] Hang Yan, Sachini Herath, and Yasutaka Furukawa. Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods. *CoRR*, abs/1905.12853, 2019.

[22] Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501, 1990.

[23] Ilya Sutskever. *Training Recurrent Neural Networks*. PhD thesis, CAN, 2013.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith 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, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[25] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[26] Jrgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. pages 573–580, 10 2012.

[27] Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. *CoRR*, abs/1710.06451, 2017.

[28] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016.