2021 IEEE/SICE International Symposium on System Integrations (SII)
Iwaki, Fukushima, Japan. January 11 - 14, 2021

DNN-based self-attitude estimation by learning landscape information

Ryota Ozaki[1] and Yoji Kuroda[2]

Abstract— **This paper presents DNN (deep neural network)-based self-attitude estimation by learning landscape information. The network predicts the gravity vector in the camera frame. The input of the network is a camera image, the outputs are a mean vector and a covariance matrix of the gravity. It is trained and validated with a dataset of images and correspond gravity vectors. The dataset is collected in a simulator. Using a simulator breaks the limitation of amount of collecting data with ground truth. The validation showed the network can predict the gravity vector from only a single shot image. It also showed the covariance matrix expresses the uncertainty of the prediction.**

I. INTRODUCTION

Estimating attitude of a robot is one of the classic problems of mobile robotics. Especially, real-time estimation is required for real-time attitude control. The attitude is generally estimated with inertial sensors such as an accelerometer and a gyroscope. However, mobile robots have their own acceleration. Especially, on-road robots also receive pulses from the ground. These need to be filtered out from a accelerometer. On the other hand, integration of a gyroscope has the problem of drift and bias. These disturbances worsen the accuracy of the estimation. To complement each other, these inertial data are fused, generally[1]. Nevertheless, dealing the disturbances with only inertial sensors is quite difficult.

To reduce the influence of these disturbances, many kinds of SLAM (Simultaneous Localization And Mapping)[2] have been proposed. SLAM with LiDAR matches point clouds by ICP (iterative closest point)[3], NDT (normal distributions transform)[4], and so on. Many visual SLAM with cameras have also been proposed[5], [6]. SLAM often contains accumulative error since relative pose changes with error are summed up. In order to correct the accumulative error, prior information such as 3D maps is used[7]. These methods correct the error by matching the prior information against data from the sensor. However, they work only in environment which its map is available. Moreover, creating a map is time-consuming and requires update. Some methods[8], [9] estimate attitude under Manhattan world assumption. They assume that planes and edges in an environment are orthogonal to each other. It helps achieving drift-free estimation. It is difficult for this kind of methods to avoid being affected by objects which do not satisfy assumption.

Deep learning has been used for attitude estimation in recent years. In [10], IMU-based odometry by end-to-end learning have been proposed. In [11], a deep neural network identifies the measurement's noise characteristics of IMU. In [12], a neural network estimates angular rates from sequential images. It was trained with synthetic and real images. The large synthetic dataset was collected in AirSim[13] which offers visually realistic graphics. In [14], a gravity vector is directly estimated from a single shot image. This is based on expectation that the network can learn edge information, context information, and landscape information; for example, most artificial buildings should be built vertically, the sky should be seen when the camera orients upper, and so on. The method does not depend time sequence since only a single shot image is used to estimate attitude. It helps suppressing drift, noise, and accumulative error. This method is the most similar to our proposed method. However, this method contains some problems. It cannot express uncertainty of the prediction; for instance, the network outputs estimation even when the camera is covered by objects, when less features are captured, and so on. These outputs with large error worsen estimation when a filter function such as Kalman filter[16] combines some methods. Therefore they need to be rejected.

To address these issues above, this paper presents self-attitude estimation with DNN which predicts a gravity vector from a single shot image, where the outputs are mean and covariance. The differences from the related work[14] are noted below:

- A larger dataset is collected in our work by using a simulator.
- Our network is trained with reliable ground truth while the related work collects a dataset with hand carried phone.
- Our network applies L2 normalization to the output gravity vector while ReLU[15] is applied in the related work.
- Our network outputs mean and covariance matrix while the related work directly outputs the gravity vector.

The source code used in this paper for collecting the dataset and for deep learning have been released in open repositories.

II. DNN-BASED SELF-ATTITUDE ESTIMATION BY LEARNING LANDSCAPE INFORMATION

The proposed method makes a deep neural network learn landscape information for estimating a gravity vector in a camera frame. The gravity vector is expressed as mean and covariance to consider uncertainty of the prediction.

A. Coordinate definition

A world frame is defined as a standard right-handed coordinate system. A camera frame is defined as a right-

[1]Ryota Ozaki is with Graduate School of Science and Technology, Meiji University, Kanagawa, 214-8571, Japan ce192021@meiji.ac.jp

[2]Yoji Kuroda is with Graduate School of Science and Technology, Meiji University, Kanagawa, 214-8571, Japan ykuroda@meiji.ac.jp

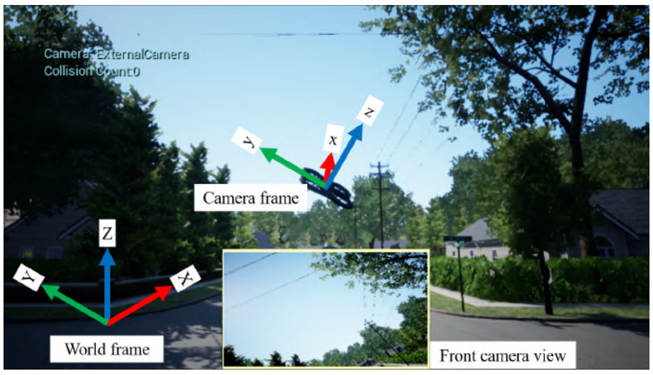


Fig. 1: Screenshot of AirSim with coordinate description. An IMU and a camera are equipped to the drone in the simulator. A purpose of this work is estimating a gravity vector in the camera frame from a front camera image.

handed coordinate system which is fixed on the camera pose. They are shown in Fig.1.

B. Dataset collection

A dataset is collected in AirSim[13]. AirSim is a simulator for drones, cars and more, built on Unreal Engine, which provides visually realistic graphics. The dataset consists of images and correspond gravity vectors $\boldsymbol{g}$ in the camera frame. The camera pose and weather parameters are randomized, and a image and a gravity vector are recorded at each pose. The range of random Z is limited as [2 m, 3 m] in this work. The ranges of random roll ϕ and pitch θ are limited as [-30 deg, 30 deg], respectively. Fig.2 shows examples of the dataset.

C. Data transformation and augmentation

A input data and a label data are transformed and are augmented in each epoch of training.

1) Image (input): A image is randomly rotated for augmenting data. The rotation angle α is limited as [-10 deg, 10 deg]. The image is resized to 224×224. RGB values are normalized. In this work, this normalization is done following $\boldsymbol{mean} = (0.5, 0.5, 0.5)$ and $\boldsymbol{std} = (0.5, 0.5, 0.5)$. Fig.3 shows an example of data augmentation.

2) Gravity vector (label): A gravity vector in the camera frame is rotated according to α. Since the network does not need to learn norm of the gravity, L2 normalization is also applied to the vector in order to make training efficient.

$$\boldsymbol{g}_{\mathrm{trans}} = \frac{\boldsymbol{R}_{(\alpha)}\boldsymbol{g}}{|\boldsymbol{R}_{(\alpha)}\boldsymbol{g}|}, \quad \boldsymbol{R}_{(\alpha)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(-\alpha) & -\sin(-\alpha) \\ 0 & \sin(-\alpha) & \cos(-\alpha) \end{pmatrix} \tag{1}$$

D. Network

The proposed deep neural network is shown in Fig.4. It consists of CNN (convolutional neural network) layers and FC (fully connected) layers. Its input is a resized image, and its outputs are a mean vector of a gravity vector and

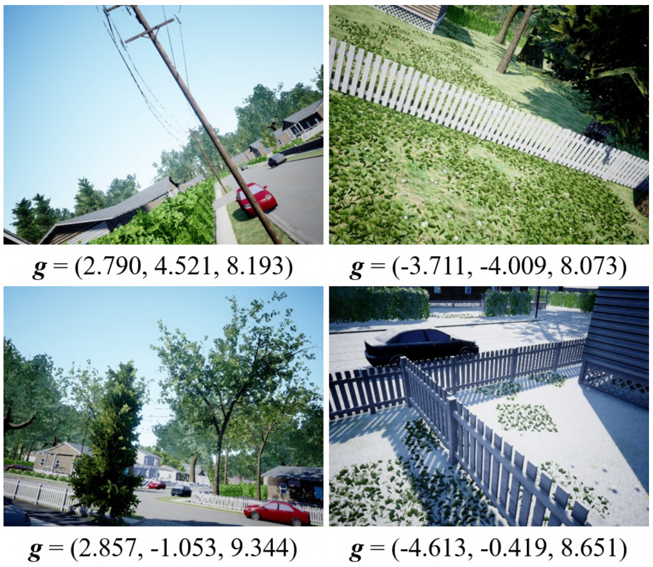

$\boldsymbol{g}$ = (2.790, 4.521, 8.193)
$\boldsymbol{g}$ = (-3.711, -4.009, 8.073)
$\boldsymbol{g}$ = (2.857, -1.053, 9.344)
$\boldsymbol{g}$ = (-4.613, -0.419, 8.651)

Fig. 2: Examples of datasets. The dataset consists of images and correspond gravity vectors $\boldsymbol{g}[\mathrm{m/s^2}]$ in the camera frame. These data are collected in "Neighborhood" of AirSim. The camera pose and weather parameters are randomized for creating a dataset. Road in the lower right image is covered by snow.

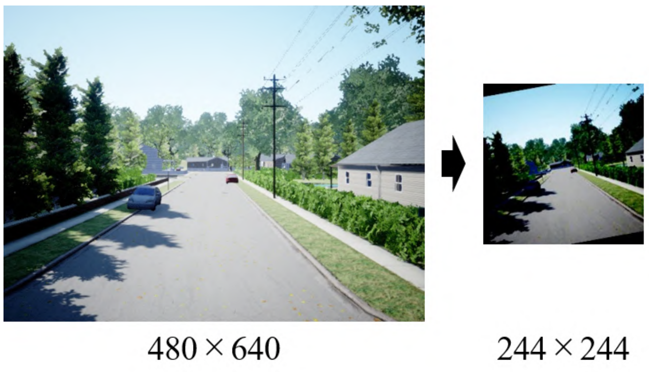


Fig. 3: Example of a transformed image. An image is randomly rotated according to α. This example shows an image when $\alpha = 10\,\mathrm{deg}$. It is also resized, and is normalized.

a covariance matrix. Technically, the output of FC layers is $(\mu_{g_x}, \mu_{g_y}, \mu_{g_z}, L_0, \cdots, L_5)$, and the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ are computed as Eq.2, 3, respectively. Since, the lower-triangular matrix $\boldsymbol{L}$ is required to have positive-valued diagonal entries, an exponential function is applied to the diagonal elements.

$$\boldsymbol{\mu} = \frac{(\mu_{g_x}, \mu_{g_y}, \mu_{g_z})^{\mathrm{T}}}{|(\mu_{g_x}, \mu_{g_y}, \mu_{g_z})^{\mathrm{T}}|} \tag{2}$$

$$\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}, \quad \boldsymbol{L} = \begin{pmatrix} \exp(L_0) & 0 & 0 \\ L_1 & \exp(L_2) & 0 \\ L_3 & L_4 & \exp(L_5) \end{pmatrix} \tag{3}$$

It is expected that the CNN layers lean extracting features such as edges, and FC layers learn landscape information.

Feature module of VGG16[17] pre-trained on ImageNet[18] is adopted as the CNN layers of the proposed method. All layers, except the final output layer, use the ReLU function[15] as activation function. All FC layers, except the final output layer, use the 10% Dropout[19].

E. Loss function

Assuming that the estimation follows multivariate normal distribution, probability density is computed as Eq.4.

$$P(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{\exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}))}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}}, \quad k = \mathrm{rank}(\boldsymbol{\Sigma}) \tag{4}$$

where k denotes dimensions of variables, i.e. $k = 3$ in the proposed method. To make the network learn the probabilistic model, it is trained maximizing $P(\boldsymbol{x}_{\mathbf{label}}|\boldsymbol{\mu},\boldsymbol{\Sigma})$, where $\boldsymbol{x}_{\mathbf{label}}$ (= $\boldsymbol{g}_{\mathbf{trans}}$) is a label of the dataset. The total probability density P_{total} of a dataset $D_{\mathrm{label}} = [\boldsymbol{x}_{\mathbf{label}_0},\cdots,\boldsymbol{x}_{\mathbf{label}_i},\cdots,\boldsymbol{x}_{\mathbf{label}_n}]$ is computed by multiplying as Eq.5.

$$P_{\mathrm{total}} = \prod_{i=0}^{n} P(\boldsymbol{x}_{\mathbf{label}_i}|\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i) \tag{5}$$

Since the natural logarithm is a monotonically increasing function, maximizing P_{total} can be simplified by taking the natural logarithm. This avoids the value becoming too small by multiplying, and decreases computational cost. Here, $P_{\mathrm{log_{total}}}$ denotes a total value of log-probability.

$$P_{\mathrm{log_{total}}} = \sum_{i=0}^{n} \ln P(\boldsymbol{x}_{\mathbf{label}_i}|\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i) \tag{6}$$

Moreover, maximizing $P_{\mathrm{log_{total}}}$ is equivalent to minimizing $-P_{\mathrm{log_{total}}}$. Finally, the loss function of the proposed method is Eq.7.

$$f_{(\boldsymbol{w})} = -P_{\mathrm{log_{total}}} \tag{7}$$

where $\boldsymbol{w}$ denotes parameters of the network. The network minimizes the loss by updating $\boldsymbol{w}$.

F. Optimization

Adaptative Moment Estimation (Adam)[20] is used to optimize the parameters. The learning rates are set as $lr_{\mathrm{CNN}} = 0.00001$, $lr_{\mathrm{FC}} = 0.0001$, where lr_{CNN} is a value for CNN layers, lr_{FC} is a value for FC layers.

III. VALIDATION

A. Comparative methods

Definition of methods which were used in this validation are summarized here.

1) MLE (ours, all): "MLE (ours, all)" denotes the proposed method described in Sec.II. MLE is short for Maximum Likelihood Estimation.

TABLE I: Loss after 200 epochs.

	Train	Test
MLE (ours) [m/s^2]	-6.4991	-5.7436
Regression w/ L2 normalization [m^2/s^4]	0.0011	0.0031
Regression w/o L2 normalization [m^2/s^4]	0.6588	0.4916

2) MLE (ours, selected): "MLE (ours, selected)" denotes the method uses the exactly same network and the same parameters as "MLE (ours, all)" does, but only samples which output small variance are used for validation of attitude estimation. It means samples with large variance are filtered out as outliers. Assuming β in Eq.8 expresses uncertainty of the prediction, samples with small variance are selected with a threshold TH_β. In this validation, the threshold is set as $\mathrm{TH}_\beta = \frac{1}{n}\sum_{i=0}^{n}\beta_i$, where n is a number of samples in the testing dataset.

$$\beta = \sqrt{\Sigma_{0,0}} \times \sqrt{\Sigma_{1,1}} \times \sqrt{\Sigma_{2,2}}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \Sigma_{0,0} & \Sigma_{0,1} & \Sigma_{0,2} \\ \Sigma_{1,0} & \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,0} & \Sigma_{2,1} & \Sigma_{2,2} \end{pmatrix} \tag{8}$$

3) Regression w/ L2 normalization: "Regression w/ L2 normalization" denotes the network which the final FC layer is difference from "MLE (ours)". It outputs a 3d vector without covariance. It is a similar architecture as [14]. L2 normalization is applied to the final layer while ReLU is applied in [14]. Mean square error (MSE) between labels and outputs is used as a loss function.

4) Regression w/o L2 normalization: "Regression w/ L2 normalization" denotes the regression network without L2 normalization. It is also require to learn norm of the gravity vector, i.e. approximately 9.8 m/s^2, in order to minimize the loss. This information is not required to estimate attitude ϕ, θ.

B. Train and validation

The network was trained with 10000 samples with a batch size of 200 samples for 200 epochs. Another 1000 samples were used for test. They were collected in "Neighborhood" of AirSim. The reason for the simulation layout is because it is a large enough environment with buildings. The training dataset and the test dataset were not mixed.

Loss values during training are plotted in Fig.5. The regression models converged much faster than the MLE model did. The regression model with L2 normalization converged a bit faster than the regression model without L2 normalization did. Table I shows loss values after 200 epoch training. It is noted that gradient was not computed with the test dataset. It is also noted a loss function of the MLE model and one of the regression models are difference.

C. Attitude estimation

Roll ϕ and pitch θ of the camera pose in the gravitational coordinate are estimated by using $\boldsymbol{\mu}$.

$$\phi = \tan^{-1}\frac{\mu_{g_y}}{\mu_{g_z}}, \quad \theta = \tan^{-1}\frac{-\mu_{g_x}}{\sqrt{\mu_{g_y}^2 + \mu_{g_z}^2}} \tag{9}$$

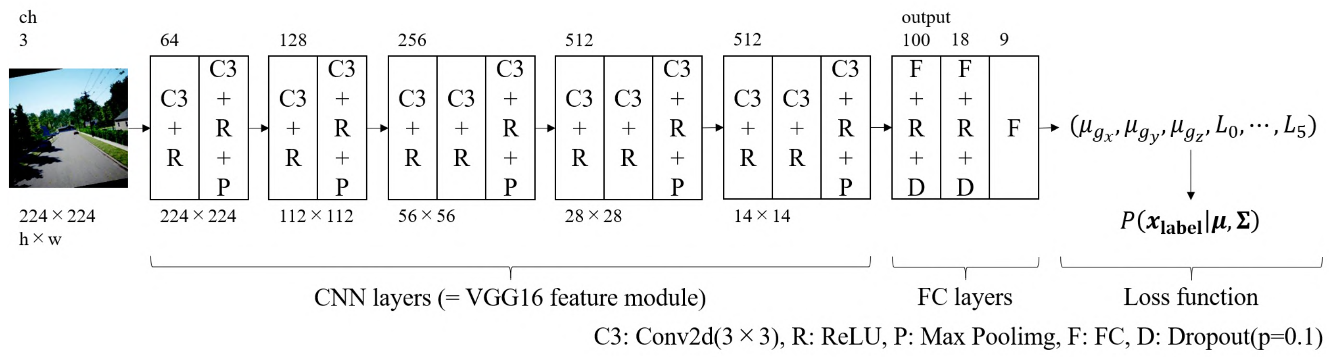
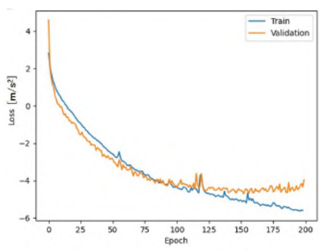
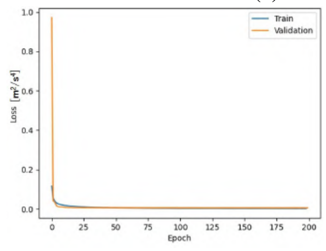

ch
3
224 × 224
h × w
64
224 × 224
128
112 × 112
256
56 × 56
512
28 × 28
512
14 × 14
output
100 18 9
C3 + R
C3 + R + P
F + R + D
F
$(\mu_{g_x}, \mu_{g_y}, \mu_{g_z}, L_0, \cdots, L_5)$
$P(\boldsymbol{x}_{\text{label}}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$
CNN layers (= VGG16 feature module)
FC layers
Loss function
C3: Conv2d(3 × 3), R: ReLU, P: Max Poolimg, F: FC, D: Dropout(p=0.1)

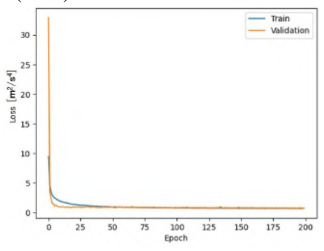
Fig. 4: Network architecture. It consists of CNN layers and FC layers. The input data is a resized image, and the output data are a mean vector and a covariance matrix. They are computed with an output from the final layer as Eq.2, 3, respectively. Log-probability of multivariate normal distribution is used as a loss function of this model.



(a) MLE (ours)



(b) Regression w/ L2 normalization



(c) Regression w/o L2 normalization

Fig. 5: Loss plotting. It is noted a loss functions of the MLE model and one of the regression models are difference. Therefore, their values can not be simply compared.

Mean absolute error (MAE) of the estimation of the validation dataset is shown in Table II. Variance of the estimated attitude error is shown in Table III. Both of the error and the variance of "MLE (ours, selected)" are smaller than the others. 715 samples which has $\beta < \mathrm{TH}_{\beta} = 0.00008814\,\mathrm{m}^3/\mathrm{s}^6$ were selected from 1000 validation samples in "MLE (ours, selected)".

Comparing "MLE (ours, all)" and "MLE (ours, selected)", filtering with TH_{β} is valid, which means the network expresses uncertainty by outputting a covariance matrix. In order to see this, the samples are sorted in Fig.6. In Fig.6a, top 50 samples are shown in descending order of the error in "MLE (ours)". In Fig.6b, the top 50 samples are shown in descending order of β in "MLE (ours)". In Fig.6a, most of the sample images with large error are covered by objects, and the images are dark with much less landscape information. It implies estimating gravity direction with much less landscape information is difficult for the DNN, just like for human. There is no way to detect them by the regression model. 21 samples of the top 50 samples are mutual of both groups. Correlation between error and β are not complete, but many samples with large error were detected by sorting samples with β. A good example with large β and one with small β are shown in Fig.7, respectively. Obviously, the sample in Fig.7a does not have enough landscape information to estimate the gravity vector, and the proposed network expresses the uncertainty with large β.

TABLE II: MAE of estimated attitude.

	Roll [deg]	Pitch [deg]
MLE (ours, all)	2.620	2.277
MLE (ours, selected)	1.836	1.467
Regression w/ L2 normalization	2.727	2.525
Regression w/o L2 normalization	2.766	2.366

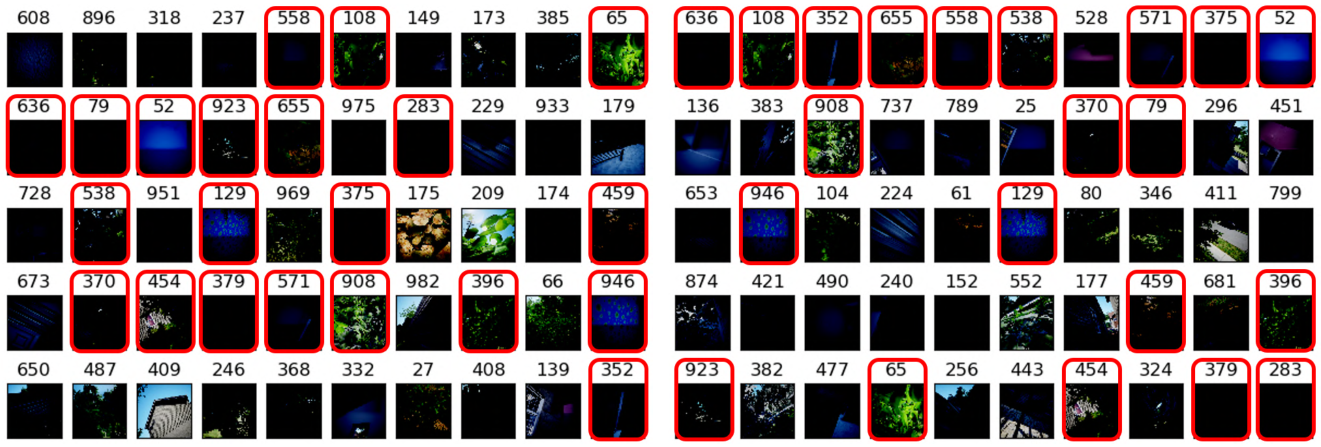
TABLE III: Variance of estimated attitude error.

	Roll [deg^2]	Pitch [deg^2]
MLE (ours, all)	23.139	18.348
MLE (ours, selected)	9.657	4.553
Regression w/ L2 normalization	25.161	21.996
Regression w/o L2 normalization	21.668	18.213

Comparing "Regression w/ L2 normalization" and "Regression w/o L2 normalization", L2 normalization does not contribute to the accuracy, although the one with L2 normalization converged a bit faster than the one without L2 normalization did.

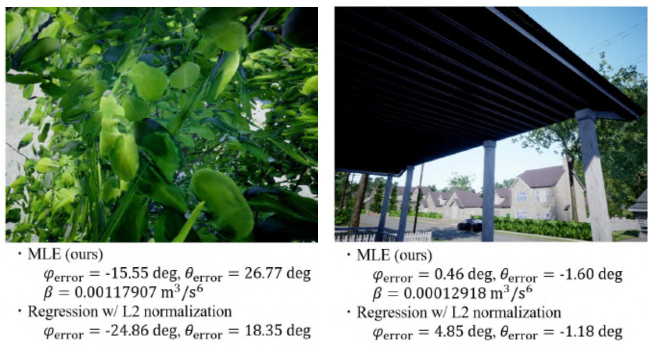
IV. Conclusions and Future Work

DNN-based self-attitude estimation by learning landscape information was proposed. A gravity vector is estimated from a single shot image. The network outputs not only the gravity vector, but also a covariance matrix. Training and validation were done with a dataset collected with AirSim. In the validation, many samples with large error are filtered out by judging variance values. It means the proposed



(a) Top 50 samples with large error in "MLE (ours)" (b) Top 50 samples with large β in "MLE (ours)"

Fig. 6: Sorted samples. A number above each image is a index of a sample. In (a), top 50 samples are sorted in descending order of the error in "MLE (ours)". Error of sample#608 in the regression model is $\phi_{\text{error}} = -35.34\,\text{deg}, \theta_{\text{error}} = -25.74\,\text{deg}$. Error of sample#352 is $\phi_{\text{error}} = 2.80\,\text{deg}, \theta_{\text{error}} = -12.93\,\text{deg}$. In (b), top 50 samples are sorted in descending order of β in "MLE (ours)". Mutual samples of the both groups are marked with red rectangles.


· MLE (ours)
φ_{error} = -15.55 deg, θ_{error} = 26.77 deg
β = 0.00117907 m^3/s^6
· Regression w/ L2 normalization
φ_{error} = -24.86 deg, θ_{error} = 18.35 deg
· MLE (ours)
φ_{error} = 0.46 deg, θ_{error} = -1.60 deg
β = 0.00012918 m^3/s^6
· Regression w/ L2 normalization
φ_{error} = 4.85 deg, θ_{error} = -1.18 deg

(a) Large β example (b) Small β example

Fig. 7: An example with large β and an example with small β. Obviously, it is hard to estimate the gravity direction from the sample(a). The proposed network expresses uncertainty of the prediction of the sample(a) by outputting large covariance.

network expresses uncertainty of the prediction by outputting a covariance matrix.

In future work, this covariance matrix may be used when the prediction is observed in Kalman filter and so on. Training and testing with datasets collected in other environments is also future work. To have good result in unknown environments, wider variety of datasets are needed. Applying the proposed method to real data is another future work. Simulator data can be used for pre-training before fine tuning with the real data.

APPENDIX

- Source code for deep learning. It is implemented using Python and PyTorch API.
  `https://github.com/ozakiryota/image_to_gravity/tree/486e2eee7e0fa8a928b45ba06c3a83cf7519c040`
- Source code for collecting dataset. It is implemented using C++ and AirSim API.
  `https://github.com/ozakiryota/airsim_controller`

ACKNOWLEDGMENT

I have received generous support from New Energy and Industrial Technology Development Organization (NEDO) for this study. I would like to thank it.

REFERENCES

[1] J. Vaganay, M. J. Aldon and A. Fournier, Mobile robot attitude estimation by fusion of inertial data, Proceedings of 1993 IEEE International Conference on Robotics and Automation (ICRA), pp.277–282, 1993.
[2] S. Thrun, W. Burgard and D. Fox, probabilistic robotics, The MIT Press, pp.309–336, 2005.
[3] S. Rusinkiewicz and M. Levoy, Efficient Variants of the ICP Algorithm, Proceedings of Third International Conference on 3-D Digital Imaging and Modeling, pp.145–152, 2001.
[4] P. Biber and W. Straßer, The Normal Distributions Transform: A New Approach to Laser Scan Matching, Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2003.
[5] J. Engel, J. Stueckler and D. Cremers: LSD-SLAM: Large-Scale Direct Monocular SLAM, Proceedings of European Conference on Computer Vision (ECCV), pp.834–849, 2014.
[6] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, ORB-SLAM: A Versatile and Accurate Monocular SLAM System, IEEE Transactions on Robotics, Vol.31, No.5, pp.1147–1163, 2015.
[7] M. A. Quddus, W. Y. Ochieng and R. B. Noland, Current map-matching algorithms for transport applications: State-of-the art and future research directions, Transportation Research Part C: Emerging Technologies, Vol.15, No.5, 312–328, 2007
[8] P. Kim, B. Coltin and H. J. Kim, Linear RGB-D SLAM for Planar Environments, Proceedings of European Conference on Computer Vision (ECCV), pp.333–348, 2018.
[9] M. Hwangbo and T. Kanade, Visual-inertial UAV attitude estimation using urban scene regularities, Proceedings of 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 2451–2458, 2011.
[10] J. P. Silva do Monte Lima, H. Uchiyama and R. I. Taniguchi, End-to-End Learning Framework for IMU-Based 6-DOF Odometry, Sensors 2019, Vol.19, No.17, 3777, 2019.

[11] M. K. Al-Sharman, Y. Zweiri, M. A. K. Jaradat, R. Al-Husari, D. Gan and L. D. Seneviratne, Deep-Learning-Based Neural Network Training for State Estimation Enhancement: Application to Attitude Estimation, IEEE Transactions on Instrumentation and Measurement, Vol.69, No.1, pp.24–34, 2020.

[12] M. Mérida-Floriano, F. Caballero, D. Acedo, D. García-Morales, F. Casares and L. Merino, Bioinspired Direct Visual Estimation of Attitude Rates with Very Low Resolution Images using Deep Networks, Proceedings of 2019 IEEE International Conference on Robotics and Automation (ICRA), pp.5672–5678, 2019.

[13] S. ShahEmail, D. DeyChris and L. Kapoor, AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles, Field and Service Robotics, Vol.5, pp.621–635, 2017.

[14] G. Ellingson, D. Wingate and T. McLain, Deep visual gravity vector detection for unmanned aircraft attitude estimation, Proceedings of 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.

[15] V. Nair and G. E. Hinton, Rectified linear units improve restricted boltzmann machines, Proceedings of ICML 2010, pp.807–814, 2010.

[16] R. E. Kalman, A new approach to linear fi ltering and prediction problems, Journal of Basic Engineering, Vol.82, pp.35–45, 1960.

[17] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2014.

[18] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, ImageNet: A large-scale hierarchical image database, Proceedings of 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 248-255, 2009.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research, vol.15, no.1, pp.1929–1958, 2014.

[20] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, Proceedings of the 3rd International Conference for Learning Representations (ICLR), 2015.