Bulletin of the JSME
日本機械学会論文集
Transactions of the JSME (in Japanese)
Vol.87, No.903, 2021

視覚画像と環境の形状を利用するリアルタイム自己姿勢推定

尾崎 亮太[*1]，黒田 洋司[*2]

Real-time self-attitude estimation using visual images and/or structures of the environments

Ryota OZAKI[*1] and Yoji KURODA[*2]

[*1,*2] Graduate School of Science and Technology, Meiji University
1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa 214-8571, Japan

Received: 18 March 2021; Revised: 27 May 2021; Accepted: 6 October 2021

Abstract

This paper presents a real-time self-attitude estimation method which utilizes the clues to the direction of the gravity hidden in images and structures of the environments. In the proposed method, the angular velocity is integrated using a gyroscope, a camera-based method estimates the gravity direction, and a LiDAR-based one also estimates the gravity direction, respectively. These estimations are integrated using the EKF (extended Kalman filter). The camera-based gravity direction estimation uses a DNN (deep neural network) which learns the regularity between the gravity direction and the landscape information. By learning the regularity, the proposed DNN can infer the gravity direction from only a single shot image. The DNN outputs the mean and variance to express uncertainty of the inference. The LiDAR-based gravity direction estimation extracts vertical planes from the surrounding environment measured by the LiDAR, and outputs the gravity direction based on their normals. By using both the camera and the LiDAR, more robust and accurate estimation can be achieved. To show the DNN can estimate the direction of gravity with uncertainty expression, static validations on test datasets are performed. Dynamic validations are also performed to show the proposed EKF-based method can estimate the attitude in real time. These validations are performed in both simulator and real world to compare the proposed method with conventional methods.

Keywords : Self-attitude estimation, Mobile robotics, Deep learning, Extended Kalman filter, Real-time estimation

1. 緒　　言

移動ロボットにおいて，ロボットの自己姿勢推定は典型的な問題の一つである．特に，姿勢を制御するためには，リアルタイムでの姿勢推定が必要である．姿勢推定は，一般的に，加速度センサや角速度センサなどの慣性センサを用いて推定される．しかし，加速度センサの計測値には，移動ロボット自身の加速度も含まれる．さらに，車輪型ロボットは地面からの振動の影響を，UAV は自身のマルチロータの振動の影響をそれぞれ受ける．これらのノイズは加速度センサから除去される必要がある．一方，角速度の積分には，ドリフトやバイアスの問題がある．上記のノイズやドリフト，バイアスは推定精度を悪化させる．そこで，加速度センサと角速度センサは，互いに補完するために統合されることが多い (Vaganay et al., 1993)．しかし，慣性センサだけでこれらの問題を扱うのは非常に難しい．

No.21-00098 [DOI:10.1299/transjsme.21-00098], J-STAGE Advance Publication date : 18 October, 2021
[*1] 学生員，明治大学 理工学研究科（〒214-8571　神奈川県川崎市多摩区東三田 1-1-1）
[*2] 正員，明治大学 理工学研究科
E-mail of corresponding author: ce192021@meiji.ac.jp

これらの影響を軽減するために，カメラや LiDAR を用いた様々な SLAM（simultaneous localization and mapping）(Thrun et al., 2005) が提案されている．LiDAR を用いた SLAM では，NDT（normal distribution transform）(Biber and Straßer, 2003) などを用いて，時系列の異なる点群をマッチングさせることで相対移動量を推定する．カメラを用いた SLAM は，時系列画像の特徴点などをトラックすることで相対移動量を推定することが多い (Mur-Artal et al., 2015)．しかしながら，SLAM は相対移動量を積算するため，累積誤差が発生しやすい．この累積誤差を補正するために，3 次元地図などの事前情報がよく利用される (Quddus et al., 2007)．これらの手法は，センサデータと事前情報を対応付けることで誤差を補正することができる．しかし当然ながら，これらの手法は，地図が用意された環境でのみ有効である．また，地図の作成には時間と労力がかかり，更新も必要となる．マンハッタンワールド仮説のもと姿勢を推定する手法も提案されている (Hwangbo and Kanade, 2011)．この仮説は，環境内の平面や辺が互いに直交していると仮定するものである．これにより，地図などの事前情報を用いずに，ドリフトを補正することができる．しかし，これらの手法では，仮定を満たさない物体の影響を避けることが難しい．我々は，マンハッタンワールド仮説よりも緩い拘束条件を用いた手法を先行研究 (尾崎，黒田，2019) で提案した．この手法は,「一般的な建造物が鉛直に建てられている」という規則性を利用したものである．そのため，マンハッタンワールドだけでなく，直交していない鉛直平面が存在する環境にも適用可能である．具体的な推定方法は，LiDAR 点群から鉛直な平面を抽出し，その平面の法線をもとに重力方向を推定する．Ellingson らの研究 (Ellingson et al., 2017) では，1 枚の画像から直接，重力ベクトルが推定される．この手法は，人間のように，写真を見るだけでその写真が撮影された方向を推定することができる．これは，重力方向と風景の間に規則性があることを示唆している．各時間ステップで 1 枚の画像のみが推定に用いられ，過去の時系列データには依存しないため，ドリフトが発生しない．また，慣性センサとは異なり，推定された重力ベクトルには，ロボット自身の加速度や振動は含まれない．しかし，この方法には，推論の不確かさを表現できないという問題点がある．例えば，レンズが障害物で遮られて，何も写っていない黒色の画像が入力された場合でも，根拠のない推論を出力してしまう．このような風景情報が不十分な画像が入力された場合は，検知しそれを棄却しないと推定精度が悪化する．

上記の問題を解決するために，我々は，重力方向を多変量正規分布で出力する DNN を提案した (Ozaki and Kuroda, 2021)．この先行研究では，出力された分散値に対して閾値を設けることで，誤差の大きい推論を棄却できることが示された．ただし，大きな分散が連続している間は，推定値が棄却され続けるため，姿勢が推定されない．この先行研究では，静止画像に対する静的推定にのみ着目したため，その点は問題にはならなかった．しかし，リアルタイム推定では，推定が出力されない時間は問題となる．もう一つの問題は，この先行研究ではシミュレーションデータのみを用いた評価しか行われていないことである．

この DNN を実データでのリアルタイム推定に適用するために，本論文では，実データでファインチューニング（fine-tuning）された DNN と，角速度センサを，EKF（extended Kalman filter）(Julier and Uhlmann, 1997) で統合する．角速度センサによる推定を統合することで，DNN の推論が棄却され続けている間も，提案手法は推定を高周期で出力することができる．さらに提案手法では，LiDAR を用いた重力方向推定 (尾崎，黒田，2019) も統合される.「カメラを用いた機械学習ベースの重力方向推定 (Ozaki and Kuroda, 2021)」と「LiDAR を用いたルールベースの重力方向推定 (尾崎，黒田，2019)」の両方を並行して実行する理由は以下の通りである．

- 2 つの手法を並列して用いることで，絶対姿勢を観測する機会を増やすことができる．
- 機械学習ベースの手法を用いることで，ルールベースではモデリングすることができない風景の規則性を利用することができる．
- ルールベースの手法を用いることで，DNN が学習していないデータ分布が発生した際に，それがバックアップを担うことができる．
- LiDAR を用いることで，テクスチャの有無，昼夜を問わず姿勢を推定することができる．

また，本論文で提案される「カメラを用いた機械学習ベースの重力方向推定」と「LiDAR を用いたルールベースの重力方向推定」は，先行研究と比べて，それぞれ改善点を含む．本論文による主な寄与は以下の通りである．

- 重力方向を推定する新たな 2 つの手法を EKF で統合することで，累積誤差の補正機会を増やす．
- 提案 DNN は先行研究 (Ozaki and Kuroda, 2021) と比べて以下のように改善されている．
  - 訓練時に新たなデータ水増し方法を適用する．
  - 実データに適用するために，事前学習とファインチューニングが行われる．

[DOI: 10.1299/transjsme.21-00098]

- LiDAR を用いた重力方向推定は先行研究 (尾崎，黒田，2019) と比べて以下のように改善されている．
  - 観測された各鉛直平面の信頼度を考慮するような処理を追加する．

本論文で使用したデータセットとソースコードはオープンリポジトリで公開されている[*1]．

2. 重力方向と環境の規則性を利用するリアルタイム自己姿勢推定

この章では,「角速度センサを用いた相対姿勢変化の積算」,「カメラを用いた重力方向推定」,「LiDAR を用いた重力方向推定」を EKF で統合する自己姿勢推定の手法を提案する．提案手法のシステム図を図 1 に示す．以下，2·1 節で座標系について，2·2 節でカメラを用いた重力方向推定について，2·3 節で LiDAR を用いた重力方向推定について，2·4 節で EKF による推定結果の統合について，それぞれ示す．

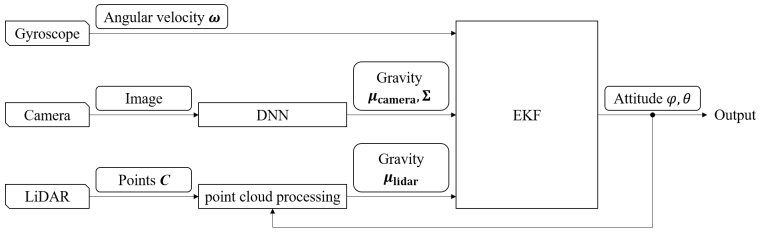


Fig. 1 Proposed EKF architecture. Gyroscopic angular rates are integrated in the prediction process in the EKF. The gravity vector estimated with a LiDAR is integrated in the update process in the EKF. The DNN outputs are integrated in another update process in the EKF.

2·1 座標系の定義

ワールド座標系は，環境に固定された右手座標系として定義され，その z 軸は鉛直上向きと一致する．ロボット座標系は，ロボットに固定された右手座標系として定義され，その x 軸はロボットの進行方向と一致する．

2·2 カメラを用いた重力方向推定

提案手法では，ロボット座標系における重力方向を推定するために，DNN に風景知識を学習させる．重力方向は推論の不確実性を考慮するために，平均と分散で表現される．

2·2·1 データセット収集

データセットは，カメラ画像と，それに対応するロボット座標系における重力ベクトルで構成される．本研究では，シミュレーションデータと実データの両方を収集する．図 2 にデータセットの例を示す．

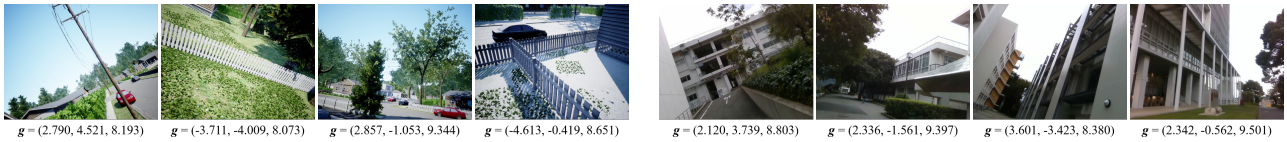
シミュレーションデータセットは AirSim(Deng et al., 2009) で収集される．AirSim は，ドローンや自動車などのシミュレータで，Unreal Engine 上に構築されている．その特徴は，視覚的にリアルなグラフィックを提供できることである．本研究では，シミュレータ内のドローンに，IMU とカメラを搭載する．ロボットの姿勢と天候パラメータをそれぞれランダムに変化させ，各姿勢での画像と加速度ベクトルを記録する．本研究では，ロボットの飛行高さ（Z 軸の範囲）は [2 m, 3 m] に限定される．同様に，ロール角 ϕ とピッチ角 θ の範囲は，それぞれ [-30 deg, 30 deg] に制限される．

実データセットは，IMU(Xsens MTi-30) とカメラ (RealSense D435) を搭載したセンサースイートを用いて収集する（図 3）．このセンサースイートを手で持ち運び，画像と加速度ベクトルを記録する．0.5 秒間でのセンサーの揺れが 0.1 deg 以下で，かつ，前ステップで記録した姿勢から 5 deg 以上離れたときにのみ，データが保存される．この条件は，データ収集時にセンサ本体に加わる不必要な振動を排除するために設定される．この IMU は仕様上，静的な状態で十分な精度（誤差 0.2 deg 以内）を有しているため，これを真値とみなす．静的な状態の IMU データを学習することで，動的な状態であっても，DNN が静的状態の IMU を再現することができる．つまり，ロ

[*1] https://github.com/ozakiryota/attitude_estimation_walls

[DOI: 10.1299/transjsme.21-00098]

ボット自身の加速度や振動を含まない加速度ベクトルが推論される．



(a) Synthetic data　　(b) Real data

Fig. 2　Examples of datasets. The dataset consists of images and corresponded gravity vectors $\boldsymbol{g}[\mathrm{m/s^2}]$ in the robot frame. The examples in (a) were collected in 'Neighborhood' of AirSim. The robot pose and weather parameters are randomized for creating the dataset. The examples in (b) were collected in the campus of Meiji University.

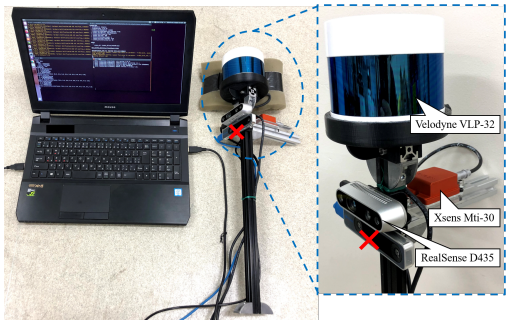


Fig. 3　Sensor suite. Images and acceleration are recorded with this sensor suite when it is still. The judge whether it is still is processed by programing. Note that depth information from RealSense D435 is not used in this study although it is a RGB-D camera.

2·2·2　データ前処理

各入力データとラベルデータには以下の前処理が適用される．さらに，学習時には毎エポックでデータ水増しを行う．図 4 に，提案するデータ水増し手法の例を示す．

- カメラデータの変換:
  各入力画像は，データを水増しするために 50%の確率で水平方向に反転される．反転処理の後，ピッチデータを水増しするために，ランダムにホモグラフィ変換が適用される．仮想のピッチ変化量 $\Delta\theta$ は [-10 deg, 10 deg] に制限される．ホモグラフィ変換後の画像の高さ h', h'' と幅 w', w'' は，それぞれ以下のように計算される．図 5 には，その変換の概略図を示す．

$$
\begin{aligned}
d &= \frac{\frac{h}{2}}{\tan(\frac{\mathrm{FOV^v}}{2})}, \quad d' = \frac{d\cos(\frac{\mathrm{FOV^v}}{2})}{\cos(\frac{\mathrm{FOV^v}}{2} - |\Delta\theta|)}, \quad d'' = \frac{\frac{h}{2}\cos(\frac{\mathrm{FOV^v}}{2} - |\Delta\theta|)}{\sin(\frac{\mathrm{FOV^v}}{2})} \\
h' &= h/2 - d\tan\left(\frac{\mathrm{FOV^v}}{2} - |\Delta\theta|\right), \quad h'' = h - h', \quad w' = \frac{d}{d'}w, \quad w'' = \frac{d}{d''}w
\end{aligned}
\tag{1}
$$

  ここで，h，w はそれぞれ元画像の高さと幅，$\mathrm{FOV^v}$ $(< 2\pi)$ はカメラの鉛直視野角（field-of-view）を表す．さらに，ロールデータを水増しするために，画像をランダムに回転させる．回転角度 $\Delta\phi$ は，[-10 deg, 10 deg] に制限される．それらの処理の後，画像は 224×224 にリサイズされる．RGB 値は，$\boldsymbol{mean} = (0.5, 0.5, 0.5)$, $\boldsymbol{std} = (0.5, 0.5, 0.5)$ に従うように標準化される．標準化を画像に適用する理由は，入力されるデータのスケールをそろえることで，学習を安定させるためである．画像の標準化とリサイズは，推論時にも適用される．我々の Python の実装によると，ホモグラフィ変換を加えることで学習時間が約 4 倍になることに注意する．ただし，変換は訓練にのみ適用されるため，推論時間には影響しない．
- IMU データの変換:

[DOI: 10.1299/transjsme.21-00098]

ラベル（正解）データの重力ベクトルも，上記の画像変換に応じて変換される．重力のノルムを学習する必要がなく，学習を効率化するために L2 正規化も適用される．

$$\hat{\boldsymbol{g}} = \begin{cases} \boldsymbol{Rot}^{\mathrm{x}}_{(-\Delta\phi)} \boldsymbol{Rot}^{\mathrm{y}}_{(-\Delta\theta)} \dfrac{\boldsymbol{g}}{|\boldsymbol{g}|} & (\mathrm{w/o\ flip}) \\ \boldsymbol{Rot}^{\mathrm{x}}_{(-\Delta\phi)} \boldsymbol{Rot}^{\mathrm{y}}_{(-\Delta\theta)} \dfrac{(g_x, -g_y, g_z)^{\mathrm{T}}}{|\boldsymbol{g}|} & (\mathrm{w/\ flip}) \end{cases} \tag{2}$$

ここで，$\boldsymbol{g}$ はデータセットに含まれるラベル（正解）の重力ベクトル，$\boldsymbol{Rot}^{\mathrm{x}}$，$\boldsymbol{Rot}^{\mathrm{y}}$ はそれぞれ，x，y 軸まわりの回転行列を表す．

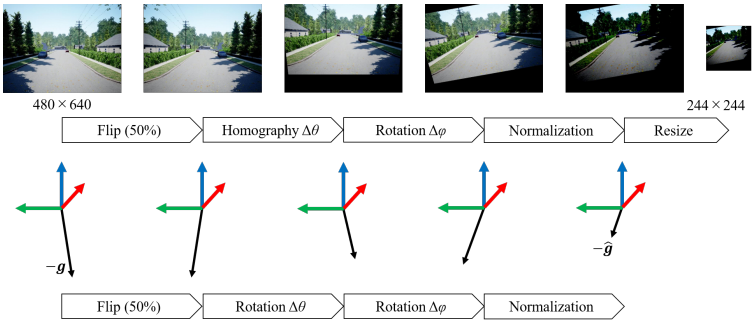


Fig. 4 Data preprocessing. Each input image is randomly flipped, transformed and rotated according to $\Delta\theta$ and $\Delta\phi$. This example shows an image when $\Delta\theta = \Delta\phi = 10$ deg. It is also resized, and is normalized. Each label vector is also transformed acording to the image preprocessing.

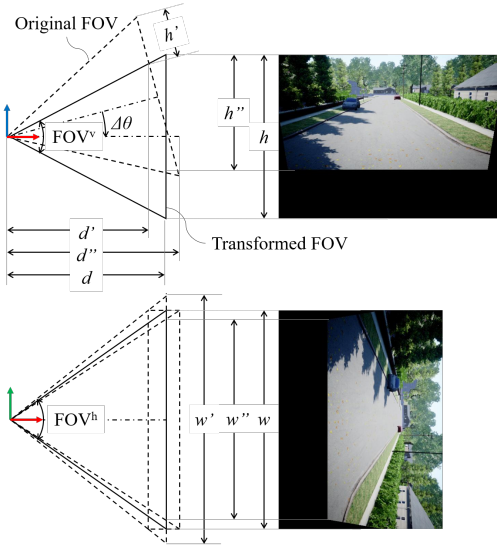


Fig. 5 Homography. The pitch data of the dataset is augmented by the homograpgy transformation.

2·2·3 ネットワーク構造

提案 DNN を図 6 に示す．ネットワークは，CNN（convolutional neural network）層と FC（fully connected）層で構成されている．ネットワークへの入力はリサイズされた画像であり，出力は重力方向の平均ベクトルと共分散行列である．厳密に言うと，FC 最終層の出力は $(\mu_x, \mu_y, \mu_z, L_0, \cdots, L_5)$ であり，平均ベクトル $\hat{\boldsymbol{\mu}}$ と共分散行列 $\boldsymbol{\Sigma}$ は，それぞれ式 3 のように計算される．共分散行列の計算に用いられる下三角行列 $\boldsymbol{L}$ は，対角成分に正の値を持つ必要があるため，それらに指数関数を適用する．

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

CNN 層はエッジなどの特徴量を抽出するために採用され，FC 層は風景知識を学習するために採用される．提案 DNN の CNN 層には，ImageNet(Deng et al., 2009) で事前学習した VGG16(Simonyan and Zisserman, 2015) のモジュールを採用する．このような転移学習は，転移するネットワークが別のタスクのために学習されたものであっても，学習を効率化することできる (Pan and Yang, 2010)．最終出力層を除く全層は，活性化関数として ReLU 関数 (Nair and Hinton, 2010) を使用する．最終出力層を除く全ての FC 層では，過学習を回避するために 10% Dropout(Srivastava et al., 2014) を使用する．

2·2·4 損失関数

提案する DNN は，深層学習を用いて最尤推定を行うことで，訓練データセットのデータ分布を学習する．それにより，DNN は入力に対する平均と分散を出力できるようになる．推論で平均 $\hat{\boldsymbol{\mu}}_t$，共分散行列 $\boldsymbol{\Sigma}_t$ が与えられた

[DOI: 10.1299/transjsme.21-00098]

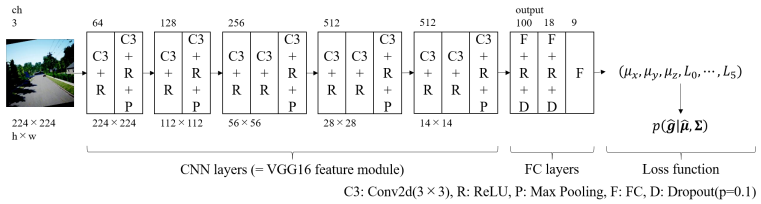


Fig. 6　Proposed network architecture. It consists of CNN layers and FC layers. The input data is a resized image, and the output data are a mean vector and a covariance matrix. They are computed with an output from the final layer as Eq.3, respectively. Log-probability of multivariate normal distribution is used as a loss function of this model.

ときの，各ラベルデータ $\hat{\boldsymbol{g}}_\iota$ に対する確率密度 $p(\hat{\boldsymbol{g}}_\iota|\hat{\boldsymbol{\mu}}_\iota, \boldsymbol{\Sigma}_\iota)$ を最大化するために，損失関数 $l_{(\boldsymbol{\Theta})}$ を以下のように設定する．

$$l_{(\boldsymbol{\Theta})} = -\sum_{\iota=0}^{\#D} \ln p(\hat{\boldsymbol{g}}_\iota|\hat{\boldsymbol{\mu}}_\iota, \boldsymbol{\Sigma}_\iota), \quad p(\hat{\boldsymbol{g}}_\iota|\hat{\boldsymbol{\mu}}_\iota, \boldsymbol{\Sigma}_\iota) = \frac{\exp(-\frac{1}{2}(\hat{\boldsymbol{g}}_\iota - \hat{\boldsymbol{\mu}}_\iota)^{\mathrm{T}}\Sigma^{-1}(\hat{\boldsymbol{g}}_\iota - \hat{\boldsymbol{\mu}}_\iota))}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}_\iota|}}, \quad d = \mathrm{rank}(\boldsymbol{\Sigma}_\iota) \tag{4}$$

ここで，$\boldsymbol{\Theta}$ はネットワークの重みパラメータ，$\#D$ はデータセットのサンプル数を表す．ディープラーニングを用いることで，上記の損失関数を最小化するように，パラメータ $\boldsymbol{\Theta}$ を更新する．

2·2·5　最適化

パラメータ $\boldsymbol{\Theta}$ の最適化には，Adam（adaptative moment estimation）(Kingma and Ba, 2015) を用いる．シミュレーションデータを用いた学習では，学習率は $lr_{\mathrm{CNN}} = 1 \times 10^{-5}$，$lr_{\mathrm{FC}} = 1 \times 10^{-4}$ とする． ここで，lr_{CNN} は CNN 層に適用される学習率，lr_{FC} は FC 層に適用される学習率である．実データを用いたファインチューニングでは，学習率はより小さく設定され，それぞれ $lr_{\mathrm{CNN}} = 1 \times 10^{-6}$，$lr_{\mathrm{FC}} = 1 \times 10^{-5}$ とする．

2·2·6　不確かさの表現

本研究では，以下の η が推論の不確かさを表すものと仮定する．この値は，分散の大きい推論を棄却するために用いられる．

$$\eta = \sqrt{\sigma_x^2} \times \sqrt{\sigma_y^2} \times \sqrt{\sigma_z^2}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z^2 \end{pmatrix} \tag{5}$$

2·3　LiDAR を用いた重力方向推定

この節では，LiDAR を用いたルールベースの重力方向推定を提案する．提案手法では，この推定結果を EKF で統合する．この節で提案される推定では,「一般的な建物が鉛直に建てられている」という規則性が利用される．この規則性に基づいて，LiDAR 点群から鉛直平面を抽出し，それらの法線方向と直交する方向を，重力方向として出力する．ただし，外れ値を除去するための条件を設定しており，環境内の全ての平面が鉛直であると仮定しているわけではない．この推定では，時々刻々のシングルスキャンのみを用いて推定を行うため，誤差の蓄積はない．この手法は，我々の先行研究 (尾崎，黒田，2019) に基づくが，本研究では観測された各鉛直平面の信頼度を考慮する点で異なる．具体的には，LiDAR 投影点の多い，または遠くに位置する平面ほど重力方向推定に影響を与える．遠くに位置する平面を重み付けする理由は，LiDAR の視野角の特性上，LiDAR からの距離が大きいほど計測範囲が広く，物体の全体像を観測しやすいからである．

2·3·1　鉛直な平面の抽出

LiDAR で取得する点群を $\boldsymbol{C}$ とする．

$$\boldsymbol{C} = \{\boldsymbol{c}_0, \cdots, \boldsymbol{c}_i, \cdots, \boldsymbol{c}_{\#\boldsymbol{C}}\}, \quad \boldsymbol{c}_i = \begin{pmatrix} c_{i,x} & c_{i,y} & c_{i,z} \end{pmatrix} \tag{6}$$

[DOI: 10.1299/transjsme.21-00098]

ここで，$\boldsymbol{c}_i$ はロボット座標系における各点の座標，$\#\boldsymbol{C}$ は点の数を表す．各クエリ点の近傍点を k-dimensional tree(Bentley, 1975) で探索する．センサから遠いほど点群の密度が小さくなるため，探索半径 r^i は，センサから遠いほど大きくなるように設定されている．

$$r^i = \alpha|\boldsymbol{c}_i| \tag{7}$$

$$\boldsymbol{B}^i = \left\{\cdots, \boldsymbol{c}_k, \cdots, \boldsymbol{c}_{\#\boldsymbol{B}^i}\right\} \tag{8}$$

ここで，α は探索半径の比例定数，$\boldsymbol{B}^i$ はクエリ点 $\boldsymbol{c}_i$ の近傍点群を表す．主成分分析 (PCA) を各近傍点群に適用し，フィッティングされた平面の集合を $\boldsymbol{N}$ とする．

$$\boldsymbol{N} = \{\boldsymbol{n}_0, \cdots, \boldsymbol{n}_i, \cdots, \boldsymbol{n}_{\#\boldsymbol{N}}\}, \quad \boldsymbol{n}_i = \begin{pmatrix} n_{i,a} & n_{i,b} & n_{i,c} & n_{i,d} \end{pmatrix} \tag{9}$$

ここで，$\#\boldsymbol{N}$ は法線フィッティングされた平面の数，$\boldsymbol{n}_i$ はロボット座標系における平面の成分を表す（i.e. $n_{i,a}x + n_{i,b}y + n_{i,c}z + n_{i,d} = 0$）．ただし，$\sqrt{n_{i,a}^2 + n_{i,b}^2 + n_{i,c}^2} = 1$ とする．平面度の高い鉛直平面を以下の条件で選定する．ここで，選定された平面の集合を $\check{\boldsymbol{N}}$（$\in \boldsymbol{N}$）とする．

$$\check{\boldsymbol{N}} = \left\{\cdots, \boldsymbol{n}_{\check{i}}, \cdots, \boldsymbol{n}_{\#\check{\boldsymbol{N}}}\right\} \tag{10}$$

- 近傍点群とそれにフィッティングされた平面との誤差が十分小さい．

$$e^{\check{i}} = \sum_{k=0}^{\#\boldsymbol{B}^{\check{i}}} |n_{\check{i},a}c_{k,x} + n_{\check{i},b}c_{k,y} + n_{\check{i},c}c_{k,z} + n_{\check{i},d}| < \mathrm{TH}_e \tag{11}$$

  ここで，$e^{\check{i}}$ は平面 $\boldsymbol{n}_{\check{i}}$ におけるフィッティング誤差， TH_e はその誤差に対する閾値を表す．この閾値によって，平面に投影された点群のみが抽出される．

- 近傍点の数が十分多い．

$$\#\boldsymbol{B}^{\check{i}} > \mathrm{TH}_{\#\boldsymbol{B}} \tag{12}$$

  ここで，$\mathrm{TH}_{\#\boldsymbol{B}}$ は近傍点の数の閾値を示す．この閾値によって，はっきりと観測されている面のみが抽出される．また，$\#\boldsymbol{B}^i$ が小さすぎると，e^i が小さくなりやすいため，この閾値は e^i を有効なものにするためのものでもある．

- 平面 $\boldsymbol{n}_{\check{i}}$ が鉛直に十分近い．これは以下のように，前ステップの重力の推定値を元に判断される．

$$\beta^{\check{i}} = \left| \cos^{-1} \frac{\boldsymbol{n}_{\check{i}} \cdot \boldsymbol{\mu}_{t-1}}{|\boldsymbol{n}_{\check{i}}||\boldsymbol{\mu}_{t-1}|} - \frac{\pi}{2} \right| < \mathrm{TH}_\beta \tag{13}$$

  ここで，$\beta^{\check{i}}$ は平面 $\boldsymbol{n}_{\check{i}}$ の法線と推定水平面がなす角度，$\boldsymbol{\mu}_{t-1}$ は時間ステップ t - 1 で推定された重力ベクトル，TH_β は角度 $\beta^{\check{i}}$ に対する閾値を表す．この閾値によって，鉛直面のみが抽出される．これにより，推定に外れ値（i.e. 鉛直でない平面）を使用することを避けることができる．これが可能な理由は，他のセンサを用いて EKF 内で姿勢を継続的に推定しているからである．言い換えると，前ステップの推定値 $\boldsymbol{\mu}_{t-1}$ にある程度の信頼性があるからである．

2·3·2 depth-Gaussian sphere の生成

選定された平面の集合 $\check{\boldsymbol{N}}$ の全ての法線の始点を，原点に移動させて点群 $\boldsymbol{S}$ を生成する．清水ら (清水，黒田，2014) はこの点群を ‘depth-Gaussian sphere’ と呼んでいる (cf. 我々の先行研究 (尾崎，黒田，2019) では Gaussian sphere(Horn, 1984) が用いられている)．図 7 に depth-Gaussian sphere の概念図を示す．単位球に点を投影する Gaussian sphere の代わりに depth-Gaussian sphere を用いる理由は，遠方の平面をより重み付けして推定に用いるためである．

$$\boldsymbol{S} = \left\{\cdots, \boldsymbol{s}_{\check{i}}, \cdots, \boldsymbol{s}_{\#\check{\boldsymbol{N}}}\right\}, \quad \boldsymbol{s}_{\check{i}} = -n_{\check{i},d}\begin{pmatrix} n_{\check{i},a} & n_{\check{i},b} & n_{\check{i},c} \end{pmatrix} \tag{14}$$

ここで，$\boldsymbol{s}_{\check{i}}$ は点群 $\boldsymbol{S}$ の各点であり，平面までの距離 $n_{\check{i},d}$ で重み付けされる．

[DOI: 10.1299/transjsme.21-00098]

2·3·3 depth-Gaussian sphere のクラスタリング

点群（depth-Gaussian sphere）$\boldsymbol{S}$ に対して，角度ベースのクラスタリングを適用する（cf. 我々の先行研究 (尾崎，黒田，2019) ではユークリッド距離ベースのクラスタリングが用いられている）．ここで，$\boldsymbol{W}$ はクラスタの集合，$\boldsymbol{w}_j$ は環境内の各平面（壁）を表す．

$$\boldsymbol{W} = \left\{\boldsymbol{w}_0, \cdots, \boldsymbol{w}_j, \cdots, \boldsymbol{w}_{\#\boldsymbol{W}}\right\}, \quad \boldsymbol{w}_j = \left\{\boldsymbol{s}_0^j, \cdots, \boldsymbol{s}_l^j, \cdots, \boldsymbol{s}_{\#\boldsymbol{w}_j}^j\right\} \tag{15}$$

式 16 を満たすとき，式 17 のように点 $\boldsymbol{s}_{\check{i}}$ はクラスタ $\boldsymbol{w}_j$ に分類される．

$$\gamma^j = \min\{\gamma_+^j, \gamma_-^j\} < \mathrm{TH}_\gamma, \quad \gamma_+^j = \cos^{-1}\frac{\boldsymbol{s}_{\check{i}} \cdot \sum_{l=0}^{\#\boldsymbol{w}_j} \boldsymbol{s}_l^j}{|\boldsymbol{s}_{\check{i}}||\sum_{l=0}^{\#\boldsymbol{w}_j} \boldsymbol{s}_l^j|}, \quad \gamma_-^j = \cos^{-1}\frac{-\boldsymbol{s}_{\check{i}} \cdot \sum_{l=0}^{\#\boldsymbol{w}_j} \boldsymbol{s}_l^j}{|-\boldsymbol{s}_{\check{i}}||\sum_{l=0}^{\#\boldsymbol{w}_j} \boldsymbol{s}_l^j|} \tag{16}$$

$$\boldsymbol{w}_j \leftarrow \boldsymbol{w}_j \cup \{\boldsymbol{s}_{\check{i}}\} \tag{17}$$

式 16 を満たさないときは， 式 18 のように点 $\boldsymbol{s}_{\check{i}}$ は新しいクラスタとして登録される．

$$\boldsymbol{W} \leftarrow \boldsymbol{W} \cup \boldsymbol{w}_{\mathrm{new}}, \quad \boldsymbol{w}_{\mathrm{new}} = \{\boldsymbol{s}_{\check{i}}\} \tag{18}$$

上記のクラスタリングによって，メンバが多い，または遠方のメンバを含むクラスタのノルム $\sum_{l=0}^{\#\boldsymbol{w}_j} \boldsymbol{s}_l^j$ は大きくなる．角度 γ^j の例を図 8 に示す．γ_+^j だけでなく γ_-^j も計算する理由は，次項でクラスターの外積を計算するときに，同一直線上のベクトルの外積を避けるためである．

上記のクラスタリング後，メンバ数が少ないクラスタは無視されるため，次項では，式 19 で定義されるクラスター $\check{\boldsymbol{W}}$ が重力ベクトルの推定に使用されることになる．

$$\check{\boldsymbol{W}} = \left\{\cdots, \boldsymbol{w}_{\check{j}}, \cdots, \boldsymbol{w}_{\#\check{\boldsymbol{W}}}\right\}, \quad \#\boldsymbol{w}_{\check{j}} > \mathrm{TH}_{\#\boldsymbol{w}} \tag{19}$$

ここで，$\mathrm{TH}_{\#\boldsymbol{w}}$ はメンバ数に対する閾値である．

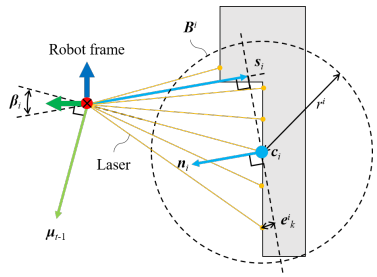


Fig. 7 Depth-Gaussian sphere. Noramls of vertical flat planes are extracted form LiDAR point cloud, and they are converted to depth-Gaussian sphere $\boldsymbol{S} = \{\cdots, \boldsymbol{s}_{\check{i}}, \cdots\}$.

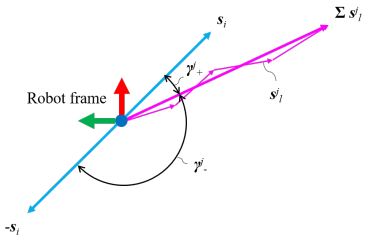


Fig. 8 Angle-based clustering. Points of depth-Gaussian sphere are clustered based on the angle γ_+^j and γ_-^j. Points with larger norm affect the direction of each cluster more.

2·3·4 鉛直面を用いた重力方向推定

ロボット座標系における重力ベクトル $\boldsymbol{\mu}$ をクラスタ $\check{\boldsymbol{W}}$ を用いて推定する．推定は，後述するように，クラスタの数 $\#\check{\boldsymbol{W}}$ に応じて，以下の各方法で計算される．

- $\#\check{\boldsymbol{W}} > 1$ の場合，クラスターの外積が重力ベクトル $\boldsymbol{\mu}$ として推定される．図 9a に計算例を示す．ノルムが大きいクラスターほど，推定に影響を与える．

$$\boldsymbol{\mu} = \sum_{\check{j}', \check{j}''(\check{j}' \neq \check{j}'')}^{\#\check{\boldsymbol{W}}} \left(\sum_{l=0}^{\#\boldsymbol{w}_{\check{j}'}} \boldsymbol{s}_l^{\check{j}'} \times \sum_{l=0}^{\#\boldsymbol{w}_{\check{j}''}} \boldsymbol{s}_l^{\check{j}''}\right) \tag{20}$$

前ステップの推定値 $\boldsymbol{\mu}_{t-1}$ と外積 $\sum \boldsymbol{s}_l^{\check{j}'} \times \sum \boldsymbol{s}_l^{\check{j}''}$ がなす角が 90 deg よりも大きい場合には，外積ベクトルを反転させる．

[DOI: 10.1299/transjsme.21-00098]

- $\#\check{\boldsymbol{W}} = 1$ の場合，前ステップの推定値 $\boldsymbol{\mu}_{t-1}$ を用いて重力ベクトル $\boldsymbol{\mu}$ を推定する．図 9b に計算例を示す．

$$\boldsymbol{\mu} = \boldsymbol{\mu}_{t-1} - \left(\boldsymbol{\mu}_{t-1} \cdot \sum_{l=0}^{\#\boldsymbol{w}_j^{\check{}}} \boldsymbol{s}_l^{\check{j}} \right) \sum_{l=0}^{\#\boldsymbol{w}_j^{\check{}}} \boldsymbol{s}_l^{\check{j}} \tag{21}$$

- $\#\check{\boldsymbol{W}} = 0$ の場合，重力ベクトル $\boldsymbol{\mu}$ は推定されない．

(a) $\#\check{\boldsymbol{W}} > 1$　　(b) $\#\check{\boldsymbol{W}} = 1$

Fig. 9　Gravity vector estimation. When the number of the clusters is more than one, the gravity vector $\boldsymbol{\mu}$ is estimated as (a). When the number of the clusters is only one, the gravity vector $\boldsymbol{\mu}$ is estimated as (b).

2·4　EKF を用いたリアルタイム姿勢推定

提案手法では，図 1 のように，角速度センサの計測値，カメラを用いた DNN の推定値（2·2 節），LiDAR を用いた点群処理による推定値（2·3 節）を，EKF で統合する．提案カルマンフィルタの状態ベクトル $\boldsymbol{x}$ は，ロボットのロール ϕ とピッチ θ で構成されている．

$$\boldsymbol{x} = \begin{pmatrix} \phi & \theta \end{pmatrix}^{\mathrm{T}} \tag{22}$$

状態ベクトル $\boldsymbol{x}$ と共分散行列 $\boldsymbol{P}$ は，予測プロセスと更新プロセスの両方で逐次的に計算される．角速度センサの計測値は，予測プロセスで積算される（2·4·1 項）．カメラを用いた DNN の推定値と，LiDAR を用いた点群処理による推定値は，それぞれ異なる更新プロセスで観測される（2·4·2，2·4·3 項）．ここで，t は時間ステップとして以下で用いられる．

2·4·1　角速度センサを用いた予測プロセス

状態ベクトル $\boldsymbol{x}$ と共分散行列 $\boldsymbol{P}$ は，それぞれ以下のように計算される．

$$\bar{\boldsymbol{x}}_t = f_{(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})} = \boldsymbol{x}_{t-1} + \boldsymbol{Rot}^{\mathrm{rpy}}_{(\boldsymbol{x}_{t-1})} \boldsymbol{u}_{t-1}, \quad \boldsymbol{u}_{t-1} = \boldsymbol{\omega}_{t-1} \Delta t = \begin{pmatrix} \omega_{\mathrm{x}_{t-1}} \Delta t \\ \omega_{\mathrm{y}_{t-1}} \Delta t \\ \omega_{\mathrm{z}_{t-1}} \Delta t \end{pmatrix} \tag{23}$$

ここで，f は状態遷移モデル，$\boldsymbol{u}$ は制御ベクトル，$\boldsymbol{\omega}$ は角速度センサで計測される 3 軸角速度，$\boldsymbol{Rot}^{\mathrm{rpy}}$ は角速度の回転行列を表す．

$$\bar{\boldsymbol{P}}_t = \boldsymbol{J}_{f_{t-1}} \boldsymbol{P}_{t-1} \boldsymbol{J}_{f_{t-1}}^{\mathrm{T}} + \boldsymbol{Q}_{t-1}, \quad \boldsymbol{J}_{f_{t-1}} = \left. \frac{\partial f}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}} \tag{24}$$

ここで，$\boldsymbol{J}_f$ は f のヤコビアン，$\boldsymbol{Q}$ はプロセスノイズの共分散行列を表す．

2·4·2　カメラを用いた更新プロセス

分散値 η が大きい場合，DNN の出力は棄却される．η の判定には閾値 TH_η が設定されており，$\eta < \mathrm{TH}_\eta$ を満たす推論のみが EKF で観測される．観測ベクトルを $\boldsymbol{z}$ とする．

$$\boldsymbol{z} = \hat{\boldsymbol{\mu}}_{\mathrm{camera}} \tag{25}$$

[DOI: 10.1299/transjsme.21-00098]

ここで，$\hat{\boldsymbol{\mu}}_{\text{camera}}$ は DNN から出力される重力の平均ベクトルを表す．観測モデルを h とする．

$$h_{(\boldsymbol{x}_t)} = \boldsymbol{Rot}^{\text{xyz}}_{(-\boldsymbol{x}_t)} \frac{\boldsymbol{g}_{\text{world}}}{|\boldsymbol{g}_{\text{world}}|}, \quad \boldsymbol{g}_{\text{world}} = \begin{pmatrix} 0 \\ 0 \\ g_{\text{world}} \end{pmatrix} \tag{26}$$

ここで，$\boldsymbol{g}_{\text{world}}$ はワールド座標系における重力ベクトル（i.e. $g_{\text{world}} \fallingdotseq 9.8\,\text{m/s}^2$），$\boldsymbol{Rot}^{\text{xyz}}$ はベクトルの回転行列を表す．観測ノイズの共分散行列 $\boldsymbol{R}$ は，DNN の出力 $\boldsymbol{\Sigma}$ を用いて設定される．

$$\boldsymbol{R} = \begin{pmatrix} \xi\sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \xi\sigma_y^2 & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \xi\sigma_z^2 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z^2 \end{pmatrix} \tag{27}$$

ここで，ξ は分散を調整するためのハイパーパラメータとする．このハイパーパラメータを導入する理由は，角速度センサ，LiDAR，カメラ DNN の出力周期が異なり，それに応じてカメラ DNN の共分散行列を調整するためである．ここで，共分散行列 $\boldsymbol{\Sigma}$ の非対角要素は各軸同士の共分散であり，$-1 \leqq \sigma_{xy} \leqq 1$，$-1 \leqq \sigma_{yz} \leqq 1$，$-1 \leqq \sigma_{zx} \leqq 1$ であるため，対角成分のみを ξ でスカラー倍する．状態ベクトル $\boldsymbol{x}$ と共分散行列 $\boldsymbol{P}$ は，それぞれ以下のように計算される．

$$\tilde{\boldsymbol{x}}_t = \boldsymbol{x}_t + \boldsymbol{K}_t(\boldsymbol{z}_t - h_{(\boldsymbol{x}_t)}), \quad \tilde{\boldsymbol{P}}_t = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{J}_{\boldsymbol{h}t})\boldsymbol{P}_t, \quad \boldsymbol{J}_{\boldsymbol{h}t} = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_t}, \quad \boldsymbol{K}_t = \boldsymbol{P}_t\boldsymbol{J}_{\boldsymbol{h}t}^{\text{T}}(\boldsymbol{J}_{\boldsymbol{h}t}\boldsymbol{P}_t\boldsymbol{J}_{\boldsymbol{h}t}^{\text{T}} + \boldsymbol{R}_t)^{-1} \tag{28}$$

ここで，$\boldsymbol{J}_{\boldsymbol{h}}$ は h のヤコビアン，$\boldsymbol{K}$ はゲイン行列，$\boldsymbol{I}$ は単位行列を示す．

2·4·3 LiDAR を用いた更新プロセス

以下の観測ベクトル $\boldsymbol{z}$ を用いて，式 26，28 と同様に，状態ベクトル $\boldsymbol{x}$ と共分散行列 $\boldsymbol{P}$ をそれぞれ更新する．

$$\boldsymbol{z} = \frac{\boldsymbol{\mu}_{\text{lidar}}}{|\boldsymbol{\mu}_{\text{lidar}}|} \tag{29}$$

ここで，$\boldsymbol{\mu}_{\text{lidar}}$ は LiDAR を用いて推定される重力ベクトルを表す．

3. 評　価　実　験

3·1 DNN の静的推定の検証

上記で提案された DNN は，訓練データセットを用いて訓練され，テストデータセットを用いて評価された．

3·1·1 手法リスト

検証に用いた手法の定義を以下でまとめる．

- ‘MLE w/o rejection’ は，2·2 節で記述されている提案 DNN を表す．MLE は Maximum likelihood estimation（最尤推定）の略である．
- ‘MLE w/ rejection (ours)’ は，‘MLE w/o rejection’ と全く同じネットワーク，パラメータを用いる．ただし，推論された分散が小さいサンプルのみを姿勢推定の検証に用いる．つまり，分散の大きいサンプルは外れ値として棄却される．式 5 の η が推論の不確かさを表すと仮定して，閾値 TH_η によって分散の小さいサンプルを選択する．本検証では，その閾値を $\text{TH}_\eta = \frac{1}{\#D}\sum_{\iota=0}^{\#D}\eta_\iota$ とする．ここで，$\#D$ はテストデータセットのサンプル数である．
- ‘Regression’ は，FC 最終層が ‘MLE (ours)’ とは異なるネットワークを表す．このネットワークは，共分散を含まない 3 次元の重力ベクトルを出力する．これは，関連研究 (Ellingson et al., 2017) に基づいて実装されている．ただし，関連研究 (Ellingson et al., 2017) は最終層に ReLU を適用しているが，ReLU は正の値しか出力しないため，ここでは L2 正規化を適用する．損失関数として，ラベルと出力の平均二乗誤差（MSE）を用いる．

[DOI: 10.1299/transjsme.21-00098]

- ‘Statistics’ はデータセットのラベル（正解）ベクトルの平均を，全サンプルに対する出力とする方法である．つまり，全サンプルの姿勢を推定するために $\sum_{i=0}^{\#D} \boldsymbol{g}_i$ を用いる．この手法の誤差を計算することは，データセットの標準偏差を計算するのと同じである．本研究では，この手法をベースラインとみなす．

3·1·2 事前学習

今回の検証で使用したデータセットを表 1 に示す．ネットワークは，10000 個のシミュレーションデータサンプル（Dataset#1）を用いて，バッチサイズを 200，エポック数を 300 として訓練された．さらに 1000 個のサンプル（Dataset#2）をテストに使用した．これらは AirSim の ‘Neighborhood’ で収集された．訓練データセットとテストデータセットは混合されない．訓練には，W-2133 CPU，Quadro GV100 GPU，32 GB RAM を搭載したコンピュータを使用した．このコンピュータを用いた訓練は約 43 時間かかった．

学習時の損失値を図 10 にプロットした．回帰モデル（‘Regression’）は，MLE モデル（‘MLE’）よりもはるかに早く収束した．表 2 に，300 エポックの学習後の損失値を示す．ただし，MLE モデルの損失関数と，回帰モデルの損失関数が異なることに注意する．

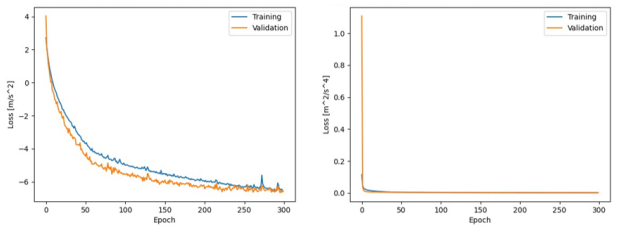
Table 1 Dataset list.

id#	Environment		#samples	Usage
1	Sim.	AirSim’s	10000	Pre-training
2		‘Neighborhood’	1000	Test
3	Real	Area-I	1108	Fine-tuning
4		Area-II (daytime)	443	Test
5		Area-II (nighttime)	447	Test

3·1·3 ファインチューニング

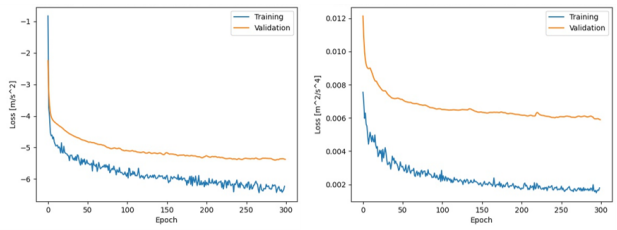
上記の事前学習の後，実データを用いたファインチューニングを行った．ネットワークは，1108 個の実データサンプル（Dataset#3）を用いて，バッチサイズを 200，エポック数を 300 として訓練された．さらに 890 個のサンプル（Dataset#4+#5）をテストに使用した．これらは明治大学のキャンパス内で収集されたものである．学習用データセットとテスト用データセットは, 同じキャンパス内で収集されたが, 同じエリアではない．また，Dataset#5 は夜間のデータである．

ファインチューニング時の損失値を図 11 にプロットした．表 3 に，300 エポックのファインチューニング後の損失値を示す．実データでの損失値は，ファインチューニングにより小さくなった．しかし，テストデータセットでの損失値は訓練データセットでの損失値よりも大きくなった．訓練データとテストデータでの結果の差を小さくするためには，より多様なデータを用いて学習を行う必要がある．



(a) MLE (ours) (b) Regression

Fig. 10 Loss plotting of training. Note that the loss function of the MLE model and one of the regression models are difference. Therefore, their values can not be simply compared.



(a) MLE (ours) (b) Regression

Fig. 11 Loss plotting of fine-tuning. The fine-tuning made the loss values on the real data smaller.

[DOI: 10.1299/transjsme.21-00098]

Table 2　Loss after 300 epochs of training.

	Train (#1)	Test (#2)
MLE (ours) [m/s^2]	-6.5002	-6.5961
Regression [m^2/s^4]	0.0014	0.0031

Table 3　Loss after 300 epochs of fine-tuning.

	Train (#3)	Test (#4+#5)
MLE (ours) [m/s^2]	-6.2295	-4.4907
Regression [m^2/s^4]	0.0018	0.0105

3·1·4　静的姿勢推定

$\hat{\boldsymbol{\mu}}$（$=\hat{\boldsymbol{\mu}}_{\text{camera}}$）を用いて，ワールド座標系におけるロボット姿勢のロール ϕ，ピッチ θ が推定される．

$$\phi = \tan^{-1}\frac{\hat{\mu}_y}{\hat{\mu}_z}, \quad \theta = \tan^{-1}\frac{-\hat{\mu}_x}{\sqrt{\hat{\mu}_y^2+\hat{\mu}_z^2}} \tag{30}$$

シミュレーションデータセットでの推定の MAE（平均絶対誤差）を表 4 に示す．‘MLE w/ rejection (ours)’ において，1000 個のテストサンプル（Dataset#2）のうち，$\eta < \mathrm{TH}_\eta = \frac{1}{\#D}\sum_{t=0}^{\#D}\eta_t = 0.000120\,\mathrm{m}^3/\mathrm{s}^6$ を満たす 795 個のサンプルが選択された．この閾値は他のデータセットでも使用された．この閾値を用いて選択されたサンプル数を表 5 に示す．実データセットでの推定値の MAE を表 6 に示す．テストデータセットでは，‘MLE w/ rejection (ours)’ の誤差は他の手法の誤差に比べて小さい結果となった．

‘MLE w/o rejection’ と ‘MLE w/ rejection (ours)’ を比較すると，TH_η によるフィルタリングが有効であることがわかり，提案ネットワークは共分散行列を出力することで不確かさを表現していることがわかる．

先行研究 (Ozaki and Kuroda, 2021) と比較して，本論文の結果はより良いものとなった．これは，データ水増しの改善が精度向上に寄与していることを示唆している．実データの収集には時間と労力がかかるため，データ水増しは特に重要である．

夜間のデータ（Dataset#5）に着目すると，‘MLE w/o rejection’ と ‘Regression’ の誤差は，日中のデータ（Dataset#4）に比べて大きくなった．‘MLE w/ rejection (ours)’ の誤差が大きくならなかったのは，図 12 のように，街灯などの影響で不確かさが小さいサンプルが，閾値 TH_η で選択されたからだと考察できる．ただし，表 5 より，夜間のデータセット（Dataset#5）では棄却されたサンプルの数が多かった．これを補うため，提案手法はカメラだけでなく LiDAR も用いており，その有効性を次節で検証する．

Table 4　MAE of static estimation on synthetic data.

MAE [deg]	Dataset#1		Dataset#2	
	Roll	Pitch	Roll	Pitch
MLE w/o rejection	1.206	1.022	1.982	1.992
MLE w/ rejection (ours)	1.014	**0.824**	**1.368**	**1.121**
Regression	**1.012**	0.852	1.948	1.902
Statistics	15.080	14.998	15.344	14.783

Table 5　Number of samples selected by MAE w/ rejection (ours).

	Dataset#1	Dataset#2	Dataset#3	Dataset#4	Dataset#5
Before fine-tuning	8388 (83.9 %)	795 (79.5 %)	672 (60.6 %)	175 (39.5 %)	102 (22.8 %)
After fine-tuning	-	-	883 (79.7 %)	238 (53.7 %)	141 (31.5 %)

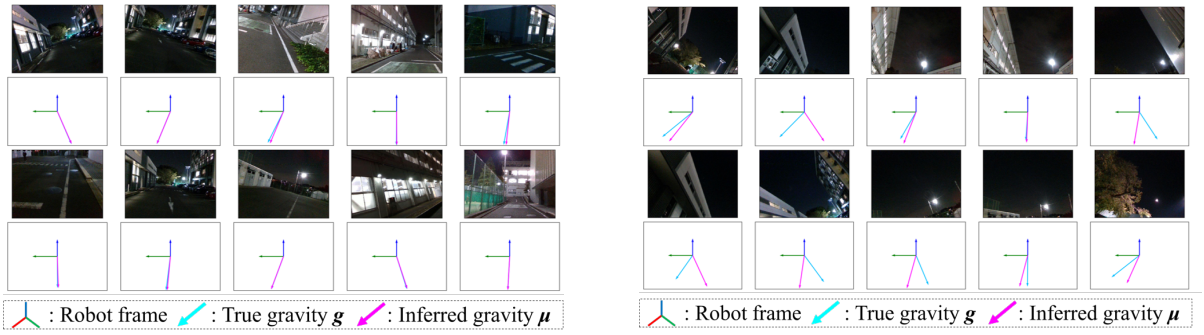
3·2　シミュレータでのリアルタイム推定の検証

シミュレータでは真値が利用可能なため，UAV の飛行シミュレーションデータを用いて，提案された EKF を用いたリアルタイム姿勢推定法を検証した．本研究は，UAV だけに着目したものではないが，様々な姿勢を再現できるため，フライトシミュレータを利用した．

[DOI: 10.1299/transjsme.21-00098]

Table 6　MAE of static estimation on real data.

MAE [deg]		Dataset#3		Dataset#4		Dataset#5	
		Roll	Pitch	Roll	Pitch	Roll	Pitch
Before fine-tuning	MLE w/o rejection	2.272	4.816	3.121	5.302	5.483	8.442
	MLE w/ rejection (ours)	**1.762**	**4.043**	**2.265**	**4.590**	**2.139**	**4.919**
	Regression	2.217	4.292	2.727	5.082	5.601	8.229
After fine-tuning	MLE w/o rejection	1.505	1.349	2.728	3.081	4.673	4.701
	MLE w/ rejection (ours)	**1.253**	**1.114**	**1.904**	**2.285**	**1.930**	**2.282**
	Regression	1.264	1.296	2.299	3.029	4.733	4.732
Statistics		15.803	10.277	15.286	13.419	19.948	15.913



(a) Top 10 smallest variance η　　(b) Top 10 largest variance η

Fig. 12　Real samples (Dataset#5) sorted in η values. The DNN tends to output large error and variance when the image is dark.

3·2·1　手法リスト

検証に用いた手法の定義を以下でまとめる．

- ‘Gyro’ は，角速度センサで計測する角速度を積算する推定手法を表す．
- ‘Gyro+Acc’ は，IMU で計測する角速度と加速度を統合する EKF ベースの推定手法を表す．
- ‘Gyro+NDT’ は，32 層の LiDAR を用いた NDT SLAM(Biber and Straßer, 2003) を表す．角速度センサで計測する角速度，真値の並進速度，NDT 出力が，EKF で統合される．ただし，真値の並進速度が利用可能なのは，環境がシミュレータであるためである．
- ‘Gyro+Regression’ は，角速度センサで計測する角速度と，回帰モデルで推論される重力ベクトルを統合する EKF ベースの推定手法を表す．ネットワークからの出力はすべて EKF に統合される．
- ‘Gyro+MLE’ は，角速度センサで計測する角速度と，提案 DNN（図 6）の推論を統合する EKF ベースの推定手法を表す．推論される分散値 η が大きい場合，推論は統合されない．ハイパーパラメータは，$\mathrm{TH}_\eta = 1.2\times10^{-4}\ \mathrm{m^3/s^6}$，$\xi = 1\times10^3$ と設定される．これらは，3·1 節の検証結果や経験を基に決められた．
- ‘Gyro+DGSphere’ は，角速度センサで計測する角速度と，LiDAR を用いた手法（2·3 節）で得られる重力ベクトルを統合する EKF ベースの推定手法を表す．‘DGSphere’ は ‘depth-Gaussian sphere’ の略である．
- ‘Gyro+MLE+DGSphere (ours)’ は 2 章で述べた本論文の提案手法を表す．処理時間を短縮するために，LiDAR 点群の 3 分の 1 の法線のみを計算する．この実験で使用したハイパーパラメータを表 7 に示す．

Table 7　Hyperparameters.

α	TH_e	$\mathrm{TH}_{\#\boldsymbol{B}}$	TH_β	TH_γ	$\mathrm{TH}_{\#\boldsymbol{w}}$	TH_η	ξ
0.09	0.05 m	10	15 deg	5 deg	20	$1.2\times10^{-4}\ \mathrm{m^3/s^6}$	1×10^3

[DOI: 10.1299/transjsme.21-00098]

3·2·2 実験条件

ドローンの飛行データは，AirSim の ‘Neighborhood’ で生成された．IMU，カメラ，LiDAR のサンプリング周期は，それぞれ約 100 Hz，5 Hz，40 Hz である．IMU の 6 軸データには仮想ノイズが付与された．そのノイズは，平均値 0 rad/s，0 m/s^2，標準偏差 0.5 rad/s, 0.5 m/s^2 の正規分布に従うようランダムに付与された．飛行コースを図 13a に示す．推定には，i7-6700 CPU，GTX1080 GPU，16 GB RAM を搭載したコンピュータを使用した．このコンピュータを用いた DNN の推論計算は 0.01-0.02 秒だった．‘DGSphere’ の計算時間は，毎ステップ約 0.1 秒だった．

(a) Simulator　　(b) Real world

Fig. 13　Driving course. The drone flew for about 9 minutes in (a). The sensors were carried for about 5 minutes in (b). The area in (b) is the same area as Area-II in Table 1.

3·2·3 実験結果

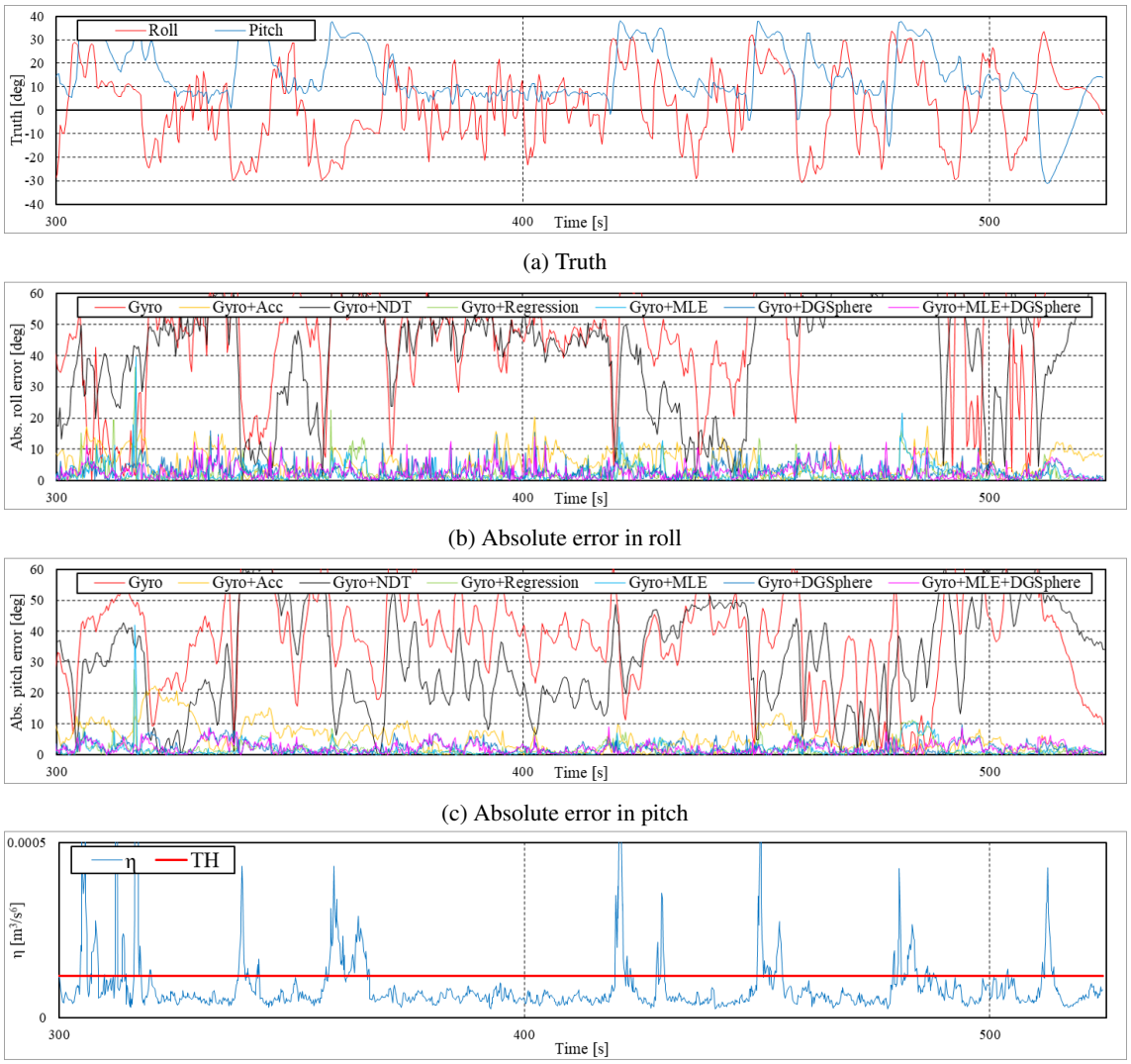
実験中に推定された姿勢を図 14 にプロットした．ただし，表示の都合上，飛行開始から 300 秒以降の結果のみを示している．表 8 に推定された姿勢の MAE を示す．‘Gyro+MLE+DGSphere (ours)’ の MAE は他の手法に比べて小さくなった．‘Gyro’ は累積誤差が大きくなった．仮想ノイズが付与されていて，他の観測がないため，これは当然のことである．‘Gyro+Acc’ は誤差を蓄積しなかった．しかし，センサの加速度値にはロボット自身の加速度やノイズが含まれているため，常に誤差が発生した．一方，提案手法では，それらを含まない重力ベクトルを観測することができる．‘Gyro+NDT’ は LiDAR を用いることで，‘Gyro’ よりゆっくり誤差を蓄積したが，累積誤差を取り除くことはできなかった．‘Gyro+Regression’，‘Gyro+MLE’，‘Gyro+DGSphere’，‘Gyro+MLE+DGSphere (ours)’ は，推定された重力ベクトルを観測することで累積誤差を補正した．‘Gyro+Regression’ と ‘Gyro+MLE’ を比較すると，η が大きい DNN の出力を棄却することが有効であることがわかった．‘Gyro+MLE’ では，飛行中に，閾値 TH_η によって約 13 %の推論が棄却された．これにより，不確実性の高い出力の観測を回避することができた．特に，‘MLE’ では，姿勢角が訓練範囲 [-30 deg, 30 deg] を超えると，分散 η が大きくなる傾向があった．一方で，推論を棄却することで，誤差を修正する機会は減る．この機会の減少を補うために，提案手法は，カメラを用いた重力方向推定と，LiDAR を用いた重力方向推定の両方を統合している．飛行中，予測プロセス（2·4·1 項）は約 51600 回，カメラを用いた更新プロセス (2·4·2 項) は約 3100 回，LiDAR を用いた更新プロセス (2·4·3 項) は約 7400 回行われた．つまり，LiDAR ベースとカメラベースの両方の推定を統合することで，累積誤差を修正する機会が増えたことを意味している．誤差を見ることで，提案された EKF でそれらを統合することが有効であることがわかった．推定の様子は，動画として公開されている[*2]．

3·3 実環境でのリアルタイム推定の検証

センサースイート（図 3）を手で持って移動し，その姿勢をリアルタイムで推定した．IMU，カメラ，LiDAR のサンプリング周期は，それぞれ約 100 Hz，15 Hz，10 Hz である．

[*2]https://photos.app.goo.gl/TZ5xkRUiTGgzsg1S8

[DOI: 10.1299/transjsme.21-00098]



(a) Truth

(b) Absolute error in roll

(c) Absolute error in pitch

(d) η

Fig. 14　Real-time plotting in ‘Neighborhood’. The graphs show the estimated error and the inferred variance in the last 225 seconds of the synthetic flight. ‘MLE’ tended to output large variance η when the angles exceed the trained range [-30 deg, 30 deg].

3·3·1　モーションキャプチャを用いた屋内実験

4.5 m × 6 m の屋内環境で約 23 分間，センサーを手で持って移動した．真値の計測には，モーションキャプチャカメラ（Vicon Vero v1.3X）を使用した．なお，DNN はこのエリアのデータで訓練されていない．

表 9 に推定姿勢の MAE を示す．提案手法は，実世界においても誤差の蓄積を抑制することができた．ただし平坦な室内環境では，提案手法の MAE は，‘Gyro+Acc’ の MAE とほぼ同じであった．振動などがより発生する環境では，‘Gyro+Acc’ の精度はより低くなると予想でき (Suwandi et al., 2019)，それは前節のシミュレーションで再現された．

3·3·2　屋外実験

モーションキャプチャカメラは姿勢を正確に測定できるが，キャプチャできる範囲が限られている．それを補うために，長距離移動実験も行った．詳細な定量評価は前項で行ったので，本項は，提案手法が実世界でも動作することを確認するためのものである．

Area-II（図 13b）で約 5 分間，センサースイートを手で持って移動した．ただし，DNN はこのエリアのデータで訓練されていない．移動中は真値が得られないため，移動終了時の推定姿勢を評価し，誤差の蓄積を確認した．

[DOI: 10.1299/transjsme.21-00098]

Table 8　MAE of dynamic estimation in simulator.

MAE [deg]	Roll	Pitch
Gyro	36.786	28.473
Gyro+Acc	6.451	5.387
Gyro+NDT	32.514	23.995
Gyro+Regression	4.317	3.071
Gyro+MLE	3.389	2.410
Gyro+DGSphere	3.288	2.407
Gyro+MLE+DGSphere (ours)	**2.772**	**2.141**

Table 9　MAE of dynamic estimation in mocap area.

MAE [deg]	Roll	Pitch
Gyro	6.012	5.100
Gyro+Acc	2.509	**1.648**
Gyro+Regression	2.843	2.730
Gyro+MLE	2.367	2.003
Gyro+DGSphere	2.365	1.878
Gyro+MLE+DGSphere (ours)	**2.242**	1.839

実験の開始時と終了時に，図 15 のように，平らな床面にセンサを置き，そのときの真値を $\phi_{\mathrm{gt}} = 0$ deg, $\theta_{\mathrm{gt}} = 0$ deg と仮定した．この評価方法は，関連研究 (Ellingson et al., 2017) に基づくものである．

表 10 に，最終姿勢での推定の誤差を示す．提案手法は，屋外走行中に，誤差の蓄積を抑制することができた．

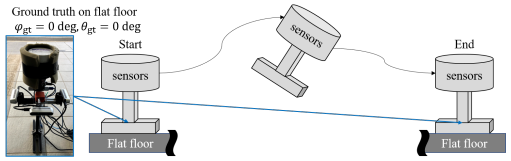


Fig. 15　Dynamic experiment. The ground truth on the flat floor is assumed to be $\phi_{\mathrm{gt}} = 0$ deg, $\theta_{\mathrm{gt}} = 0$ deg.

Table 10　Error of estimated attitude at last pose in outdoor.

Error [deg]	Roll	Pitch
Gyro+MLE+DGSphere (ours)	**+0.515**	**+2.110**
Gyro	+5.268	-5.047

3·4　提案手法の制限についての議論

本研究の制限として，提案手法には，経験的に決められるハイパーパラメータが含まれている点を言及する．特に，「LiDAR を用いたルールベースの重力方向推定」（2·3 節）では，表 7 のように，さまざまな閾値の設定が必要である．これらの閾値は，LiDAR の性能（レイヤー数，点数，視野角など）に依存して設定される．そのため，閾値を変更せずに他の LiDAR を用いる場合は，鉛直平面の抽出が適切に行われず，推定精度が低下する可能性がある．

4. 結　　言

環境中の規則性を利用する EKF ベースの自己姿勢推定法を提案した．提案手法は，「慣性センサを用いた角速度の積算」，「カメラを用いた重力方向の推定」，「LiDAR を用いた重力方向の推定」を EKF で統合する．カメラを用いた重力方向推定は，DNN を用いて，1 枚の単眼画像から重力方向を推定する．その DNN は，重力ベクトルだけでなく，共分散行列も出力する．AirSim で収集したデータセットで事前学習を行い，実センサで収集したデータセットでファインチューニングを行った．静的検証では，推論された分散値を判定することで，誤差の大きい推論が棄却された．つまり，提案された DNN は，共分散行列を出力することで推論の不確かさを表現した．その共分散行列は，EKF の統合時に観測ノイズとして用いられる．さらに，分散が大きすぎる推論は，EKF に統合される前に閾値を基に棄却される．LiDAR を用いた重力方向推定は，点群から鉛直面を抽出して，その法線方向を基に重力方向を推定する．これらのカメラベースと LiDAR ベースの重力方向推定は，ロボット自身の加速度や振動を含まない重力ベクトルを出力できる．EKF ベースの提案手法は，シミュレータと実環境の両方で検証された．

[DOI: 10.1299/transjsme.21-00098]

その動的検証では，提案手法が，カメラベースと LiDAR ベースの両方の重力方向推定を観測することで，累積誤差を補正する機会を増やし，推定誤差を抑制できることが示された．

本研究では，DNN による推論の不確かさを，標準偏差の積（式 5）で評価した．しかしながら，その他にも，行列式や，誤差楕円体の体積として評価する方法が考えられる．これらを検証するのは，今後の課題とする．また，本研究では EKF において，DNN によって出力された共分散行列を，ハイパーパラメータで恣意的に変更している（式 27）．このようなハイパーパラメータが不要な手法を検討することも，今後の課題である．

文　　献

Bentley, J. L., Multidimensional binary search trees used for associative searching, Communications of the ACM, Vol.18, No.9 (1975), pp.509–517.

Biber, P. and Straßer, W., 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), pp.2743–2748.

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

Ellingson, G., Wingate, D. and McLain, T., Deep visual gravity vector detection for unmanned aircraft attitude estimation, Proceedings of 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017), pp.5557–5563.

Horn, B. K. P., Extended gaussian images, Procceding of the IEEE, Vol.72, No.12 (1984), pp.1671–1686.

Hwangbo, M. and Kanade, T., Visual-inertial UAV attitude estimation using urban scene regularities, Proceedings of 2011 IEEE International Conference on Robotics and Automation (ICRA) (2011), pp.2451–2458.

Julier, S. J. and Uhlmann, J. K., New extension of the Kalman filter to nonlinear systems, Signal Processing, Sensor Fusion, and Target Recognition VI, Vol.3068 (1997), pp.182–193.

Kingma, D. P. and Ba, J., Adam: A method for stochastic optimization, Proceedings of the 3rd International Conference for Learning Representations (ICLR) (2015).

Mur-Artal, R., Montiel, J. M. M. and Tardós, J. D., ORB-SLAM: A Versatile and Accurate Monocular SLAM System, IEEE Transactions on Robotics, Vol.31, No.5 (2015), pp.1147–1163.

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

Ozaki, R. and Kuroda, Y., DNN-based self-attitude estimation by learning landscape information, Proceedings of 2021 IEEE/SICE International Symposium on System Integration (SII) (2021), pp.733–738.

尾崎亮太, 黒田洋司, 建造物の壁に対する相対姿勢を用いたリアルタイム 6DoF 位置姿勢推定, 日本機械学会論文集, Vol.85, No.875 (2019), pp.19–00065.

Pan, S. J. and Yang, Q., A Survey on Transfer Learning, IEEE Transactions on Knowledge and Data Engineering, Vol.22, No.10 (2010), pp.1345–1359.

Quddus, M. A., Ochieng, W. Y. and Noland, R. B., 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 (2007), pp.312–328.

Shah, S., Dey, D., Lovett, C. and Kapoor, A., AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles, Field and Service Robotics (2018), pp.621–635.

清水尚吾, 黒田洋司, 主平面を用いた点群の高速位置合わせ, 第 19 回ロボティクスシンポジア予稿集 (2014), pp.453–458.

Simonyan, K. and Zisserman, A., Very deep convolutional networks for large-scale image recognition, Proceedings of International Conference on Learning Representations (2015).

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

Suwandi, B., Kitasuka, T. and Aritsugi, M., Vehicle Vibration Error Compensation on IMU-accelerometer Sensor Using Adaptive Filter and Low-pass Filter Approaches, Journal of Information Processing, Vol.27 (2019), pp.33–40.

[DOI: 10.1299/transjsme.21-00098]

Thrun, S., Burgard, W. and Fox, D., Probabilistic Robotics (2005), pp.309–336, The MIT Press.

Vaganay, J., Aldon, M. J. and Fournier, A., Mobile robot attitude estimation by fusion of inertial data, Proceedings of 1993 IEEE International Conference on Robotics and Automation (ICRA) (1993), pp.277–282.

References

Bentley, J. L., Multidimensional binary search trees used for associative searching, Communications of the ACM, Vol.18, No.9 (1975), pp.509–517.

Biber, P. and Straßer, W., 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), pp.2743–2748.

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

Ellingson, G., Wingate, D. and McLain, T., Deep visual gravity vector detection for unmanned aircraft attitude estimation, Proceedings of 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017), pp.5557–5563.

Horn, B. K. P., Extended gaussian images, Procceding of the IEEE, Vol.72, No.12 (1984), pp.1671–1686.

Hwangbo, M. and Kanade, T., Visual-inertial UAV attitude estimation using urban scene regularities, Proceedings of 2011 IEEE International Conference on Robotics and Automation (ICRA) (2011), pp.2451–2458.

Julier, S. J. and Uhlmann, J. K., New extension of the Kalman filter to nonlinear systems, Signal Processing, Sensor Fusion, and Target Recognition VI, Vol.3068 (1997), pp.182–193.

Kingma, D. P. and Ba, J., Adam: A method for stochastic optimization, Proceedings of the 3rd International Conference for Learning Representations (ICLR) (2015).

Mur-Artal, R., Montiel, J. M. M. and Tardós, J. D., ORB-SLAM: A Versatile and Accurate Monocular SLAM System, IEEE Transactions on Robotics, Vol.31, No.5 (2015), pp.1147–1163.

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

Ozaki, R. and Kuroda, Y., DNN-based self-attitude estimation by learning landscape information, Proceedings of 2021 IEEE/SICE International Symposium on System Integration (SII) (2021), pp.733–738.

Ozaki, R., Kuroda, Y., Real-time 6DoF localization with relative poses to walls of buildings, Transactions of the JSME (in Japanese), Vol.85, No.875 (2019), pp.19–00065.

Pan, S. J. and Yang, Q., A Survey on Transfer Learning, IEEE Transactions on Knowledge and Data Engineering, Vol.22, No.10 (2010), pp.1345–1359.

Quddus, M. A., Ochieng, W. Y. and Noland, R. B., 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 (2007), pp.312–328.

Shah, S., Dey, D., Lovett, C. and Kapoor, A., AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles, Field and Service Robotics (2018), pp.621–635.

Shimizu, S. and Kuroda, Y., High-speed registration of point clouds by using dominant planes, Proceedings of the 19th Robotics Symposia (2014), pp.453–458 (in Japanese).

Simonyan, K. and Zisserman, A., Very deep convolutional networks for large-scale image recognition, Proceedings of International Conference on Learning Representations (2015).

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

Suwandi, B., Kitasuka, T. and Aritsugi, M., Vehicle Vibration Error Compensation on IMU-accelerometer Sensor Using Adaptive Filter and Low-pass Filter Approaches, Journal of Information Processing, Vol.27 (2019), pp.33–40.

Thrun, S., Burgard, W. and Fox, D., Probabilistic Robotics (2005), pp.309–336, The MIT Press.

Vaganay, J., Aldon, M. J. and Fournier, A., Mobile robot attitude estimation by fusion of inertial data, Proceedings of 1993 IEEE International Conference on Robotics and Automation (ICRA) (1993), pp.277–282.

[DOI: 10.1299/transjsme.21-00098]