

明治大学大学院 理工学研究科

2020年度

修士学位請求論文

論文題名：

風景知識を学習する DNN を用いた  
リアルタイム自己姿勢推定

指導教員名： 黒田 洋司

機械工学専攻

請求者氏名： 尾崎 亮太

2020年度 修士論文

風景知識を学習するDNNを用いた  
リアルタイム自己姿勢推定

2021年2月25日

機械工学専攻  
(学生番号: 452R192021)

尾崎 亮太

明治大学大学院理工学研究科

# 概要

この論文では、環境内の重力方向に関する規則性を利用するリアルタイム自己姿勢推定を提案する。提案する手法では、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それぞれ行う。これらの推定結果は EKF (extended Kalman filter) を用いて統合される。カメラを用いた重力方向推定は、風景知識と重力方向の規則性を学習した DNN (deep neural network) を用いる。その規則性を学習することで、提案する DNN は、1枚の単眼画像のみからその重力方向を推論する。また、提案する DNN は、推論の不確かさを表現するために、出力する重力方向を平均と分散で表現する。LiDAR を用いた重力方向推定では、LiDAR で計測された周辺環境の形状から鉛直平面を抽出し、それら平面の法線と直交する方向を重力方向として出力する。カメラと LiDAR の両方を用いることで、よりロバストで高精度な推定を実現する。DNN が、推論の不確かさとともに重力方向を推定できることを示すために、静止画像のデータセットで静的検証を行う。また、提案手法がリアルタイムで姿勢推定できることを示すために、動的検証も行われる。これらの検証は、シミュレータと実環境の両方で行われ、提案手法と従来手法を比較する。

# 目 次

<b>概要</b>	i
<b>第 1 章 序論</b>	1
<b>第 2 章 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</b>	4
2.1 座標系の定義 . . . . .	4
2.2 カメラを用いた重力方向推定 . . . . .	4
2.2.1 データセット収集 . . . . .	5
2.2.2 データ前処理 . . . . .	6
2.2.3 ネットワーク構造 . . . . .	8
2.2.4 損失関数 . . . . .	9
2.2.5 最適化 . . . . .	10
2.2.6 不確かさの表現 . . . . .	11
2.3 LiDAR を用いた重力方向推定 . . . . .	11
2.3.1 水平な法線の抽出 . . . . .	11
2.3.2 depth-Gaussian sphere の生成 . . . . .	12
2.3.3 depth-Gaussian sphere のクラスタリング . . . . .	13
2.3.4 鉛直面を用いた重力方向推定 . . . . .	14
2.4 EKF を用いたリアルタイム姿勢推定 . . . . .	15
2.4.1 角速度センサを用いた予測プロセス . . . . .	16
2.4.2 カメラを用いた更新プロセス . . . . .	16
2.4.3 LiDAR を用いた更新プロセス . . . . .	17
<b>第 3 章 評価実験</b>	18
3.1 DNN の静的推定の検証 . . . . .	18
3.1.1 手法リスト . . . . .	18
3.1.2 事前学習 . . . . .	18
3.1.3 ファインチューニング . . . . .	20
3.1.4 静的姿勢推定 . . . . .	21
3.2 シミュレータでのリアルタイム推定の検証 . . . . .	21
3.2.1 手法リスト . . . . .	22
3.2.2 実験条件 . . . . .	23

3.2.3 実験結果 . . . . .	24
3.3 実環境でのリアルタイム推定の検証 . . . . .	26
3.3.1 モーションキャプチャを用いた屋内実験 . . . . .	26
3.3.2 屋外実験 . . . . .	26
3.4 提案手法の制限についての議論 . . . . .	27
<b>第4章 結論</b>	<b>29</b>
<b>謝辞</b>	<b>30</b>
<b>付録A 公開データ</b>	<b>34</b>
<b>付録B 発表業績一覧</b>	<b>35</b>

# 図 目 次

2.1	Proposed EKF architecture.	4
2.2	Coordinate description.	5
2.3	Examples of datasets.	6
2.4	Sensor suite.	7
2.5	Data preprocessing.	8
2.6	Homography.	9
2.7	Proposed network architecture.	10
2.8	Depth-Gaussian sphere.	13
2.9	Angle-based clustering.	14
2.10	Gravity vector estimation.	15
3.1	Loss plotting of training.	19
3.2	Loss plotting of fine-tuning.	20
3.3	Synthetic samples (Dataset#2) sorted in $\eta$ values.	24
3.4	Real samples (Dataset#5) sorted in $\eta$ values.	24
3.5	Driving course.	26
3.6	Real-time plotting in ‘Neighborhood’.	27
3.7	Dynamic experiment.	28

# 表 目 次

3.1	Dataset list.	19
3.2	Loss after 300 epochs of training.	19
3.3	Loss after 300 epochs of fine-tuning.	20
3.4	MAE of static estimation on synthetic data.	22
3.5	Number of samples selected by MAE w/ rejection (ours).	22
3.6	MAE of static estimation on real data.	23
3.7	Hyperparameters.	25
3.8	MAE of dynamic estimation in simulator.	25
3.9	MAE of dynamic estimation in mocap area.	28
3.10	Error of estimated attitude at last pose in outdoor.	28

# 第1章 序論

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ー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 (e.g. SINS) [1, 2]。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。

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) [3]が提案されている。LiDARを用いたSLAMでは、ICP (iterative closest point) [4] や NDT (normal distribution transform) [5]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 [6, 7]。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。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 [8]。これらの方法は、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 [9, 10]。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 [11]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。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。

近年では、ディープラーニングが姿勢推定に利用されている。関連研究 [12]では、end-to-end学習によってIMUのみを用いたオドメトリを提案している。関連研究 [13]では、ディープニューラルネットワークを用いて、IMUの観測ノイズの特性を同定している。関

連研究[14]では、ニューラルネットワークを用いて、時系列画像から角速度を推定している。そのネットワークは、シミュレータで作成された画像と実画像を用いて訓練される。大量のシミュレーションデータは、Microsoft AirSim[15]を用いて収集された。このシミュレータは、写実的な描画を提供する。関連研究[16]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、重力方向の平均ベクトルだけでなく共分散行列も出力するDNNを提案した[17]。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、大きな分散が連續している間は、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)[18, 19]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[11]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[17]」と「LiDARを用いたルールベースの重力方向推定[11]」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 [17] 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。
- LiDAR を用いた重力方向推定は先行研究 [11] 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（付録を参照）。

以下、2章で提案手法の詳細を示し、3章でシミュレーションと実世界での実験例を示し、4章で結論および今後の課題をまとめめる。

# 第2章 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 2.1 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。

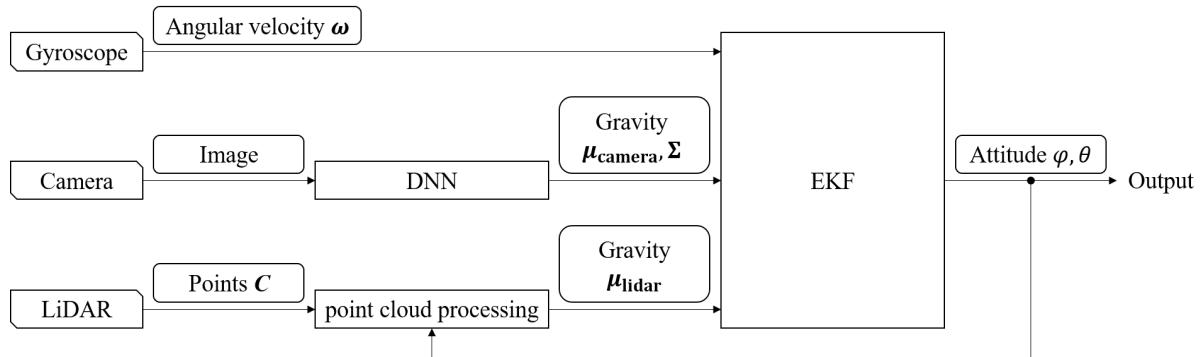


図 2.1: Proposed EKF architecture.

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。これらの座標系は図 2.2 に示される。

## 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。

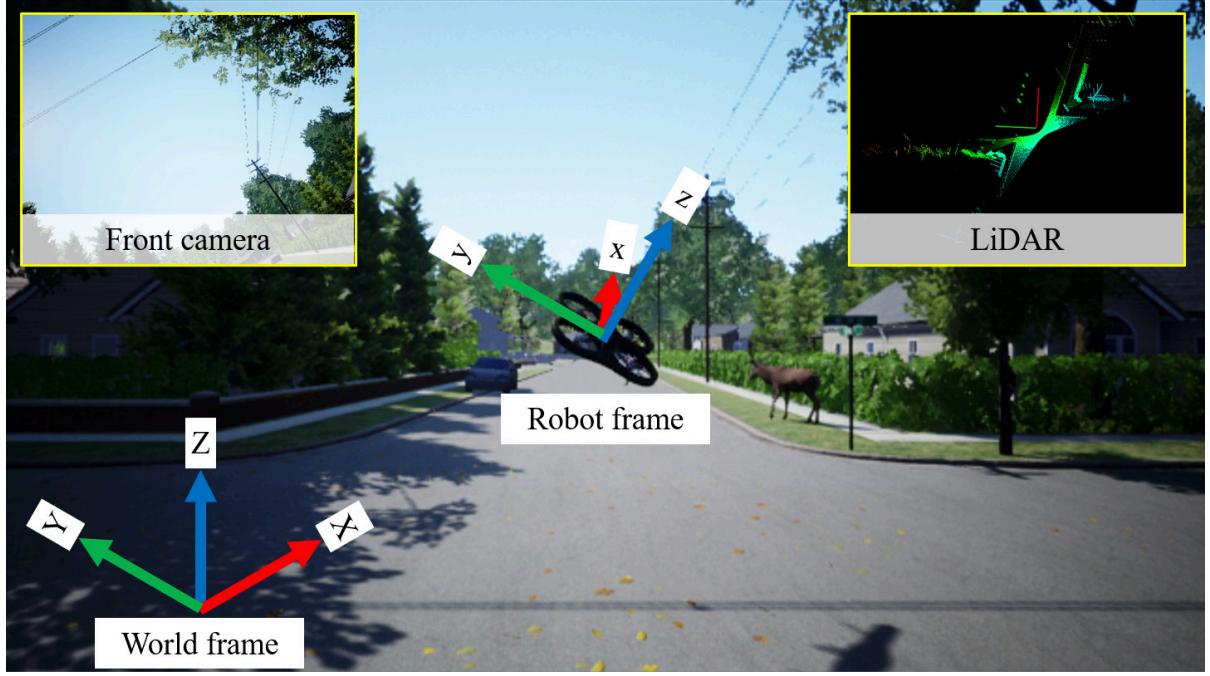


図 2.2: Coordinate description.

This study defines a robot frame and a world frame. The object of the study is estimating the attitude of the robot frame in real-time.

### 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 2.3 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[15] 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  $\phi$  とピッチ角  $\theta$  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 2.4)。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。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 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。

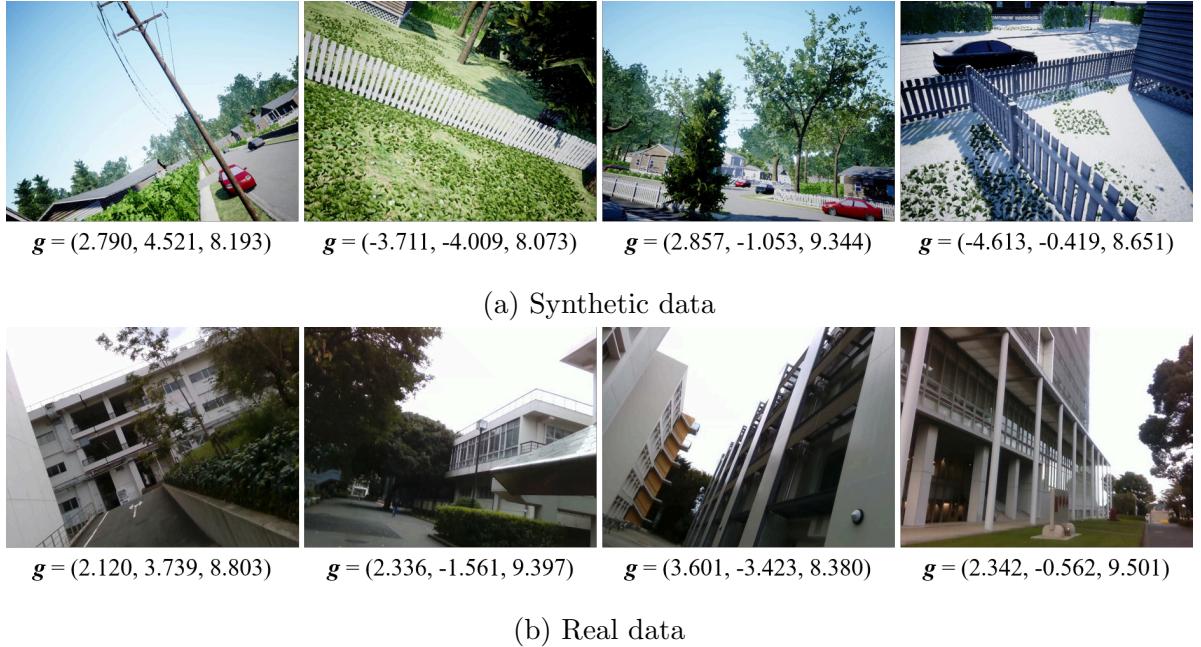


図 2.3: Examples of datasets.

The dataset consists of images and corresponded gravity vectors  $\mathbf{g}$  [m/s<sup>2</sup>] 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.

## 2.2.2 データ前処理

各入力データとラベルデータには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 2.5 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 2.6 には、その変換の概略図を示す。

$$\begin{aligned}
 d &= \frac{\frac{h}{2}}{\tan(\frac{\text{FOV}^v}{2})}, & d' &= \frac{d \cos(\frac{\text{FOV}^v}{2})}{\cos(\frac{\text{FOV}^v}{2} - |\Delta\theta|)}, & d'' &= \frac{\frac{h}{2} \cos(\frac{\text{FOV}^v}{2} - |\Delta\theta|)}{\sin(\frac{\text{FOV}^v}{2})} \\
 h' &= h/2 - d \tan\left(\frac{\text{FOV}^v}{2} - |\Delta\theta|\right), & h'' &= h - h' \\
 w' &= \frac{d}{d'}w, & w'' &= \frac{d}{d''}w
 \end{aligned} \tag{2.1}$$

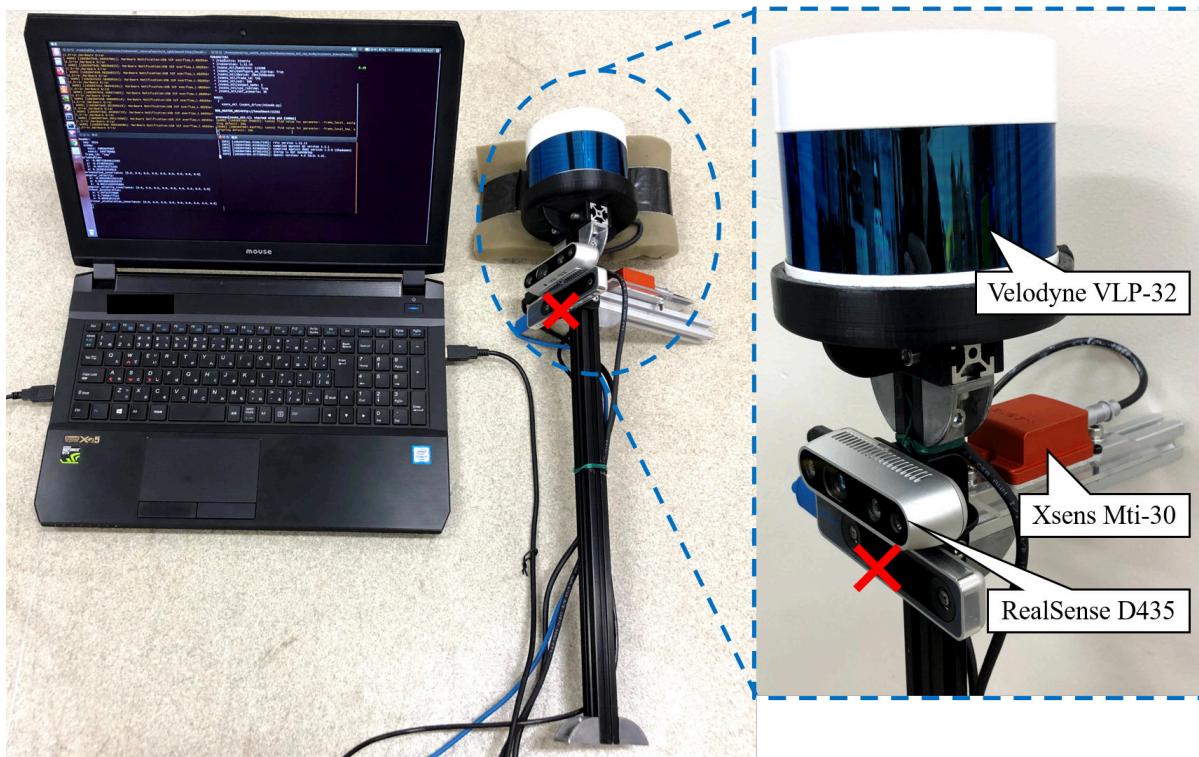


図 2.4: Sensor suite.

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.

ここで,  $h$ ,  $w$  はそれぞれ元画像の高さと幅,  $\text{FOV}^v (< 2\pi)$  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,  $[-10 \text{ deg}, 10 \text{ deg}]$  に制限される. それらの処理の後, 画像は  $224 \times 224$  にリサイズされる. RGB 値は,  $\text{mean} = (0.5, 0.5, 0.5)$ ,  $\text{std} = (0.5, 0.5, 0.5)$  に従うように正規化される. 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.

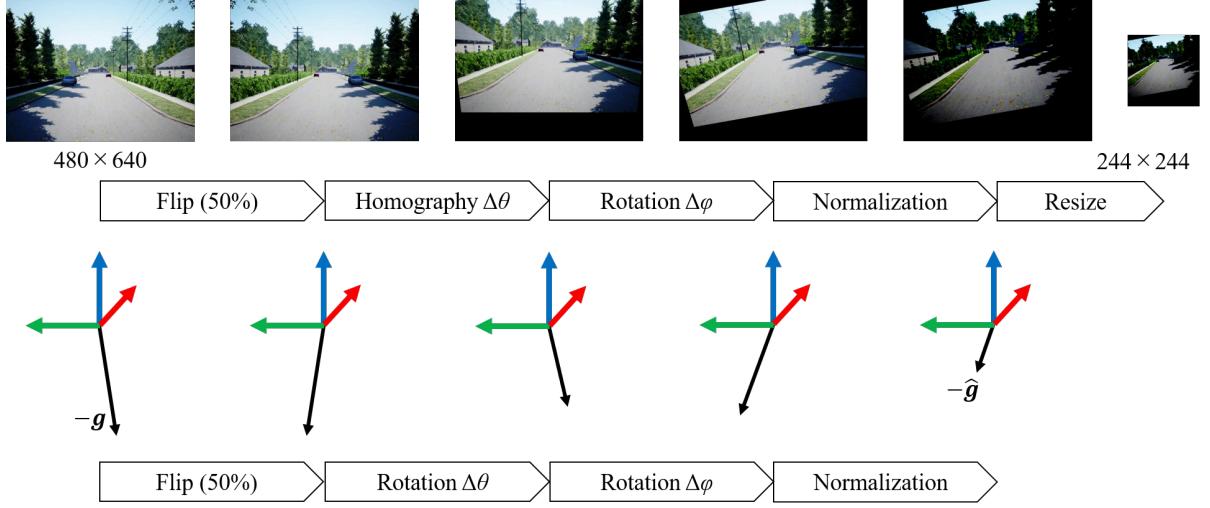


図 2.5: Data preprocessing.

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 according to the image preprocessing.

### IMU データの変換

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。

$$\hat{\mathbf{g}} = \begin{cases} \mathbf{Rot}_{(-\Delta\phi)}^x \mathbf{Rot}_{(-\Delta\theta)}^y \frac{\mathbf{g}}{|\mathbf{g}|} & (\text{w/o flip}) \\ \mathbf{Rot}_{(-\Delta\phi)}^x \mathbf{Rot}_{(-\Delta\theta)}^y \frac{(g_x, -g_y, g_z)^T}{|\mathbf{g}|} & (\text{w/ flip}) \end{cases} \quad (2.2)$$

$$\mathbf{Rot}_{(\Delta\phi)}^x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\Delta\phi) & -\sin(\Delta\phi) \\ 0 & \sin(\Delta\phi) & \cos(\Delta\phi) \end{pmatrix}, \quad \mathbf{Rot}_{(\Delta\theta)}^y = \begin{pmatrix} \cos(\Delta\theta) & 0 & \sin(\Delta\theta) \\ 0 & 1 & 0 \\ -\sin(\Delta\theta) & 0 & \cos(\Delta\theta) \end{pmatrix}$$

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

### 2.2.3 ネットワーク構造

提案 DNN を図 2.7 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  $(\mu_x, \mu_y, \mu_z, L_0, \dots, L_5)$  であり、平均ベクトル  $\hat{\mu}$  と共分散行列  $\Sigma$  は、それぞれ式 2.3, 2.4 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  $\mathbf{L}$  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。

$$\hat{\mu} = \frac{(\mu_x, \mu_y, \mu_z)^T}{|(\mu_x, \mu_y, \mu_z)^T|} \quad (2.3)$$

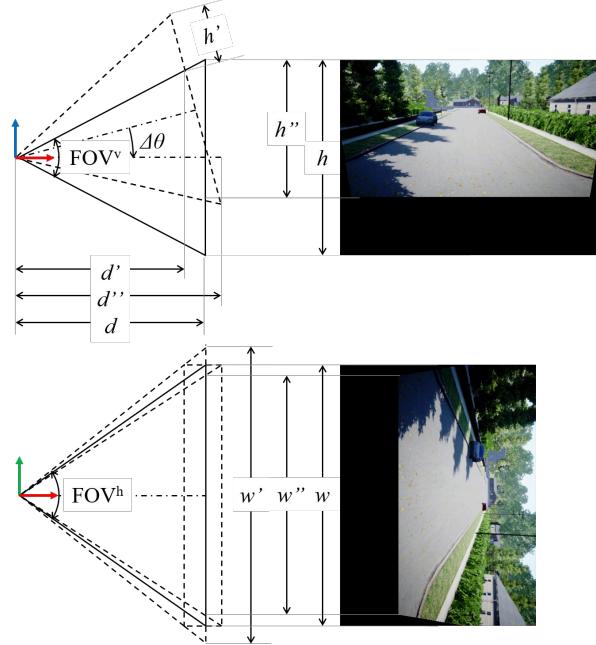


図 2.6: Homography.

The pitch data of the dataset is augmented by the homography transformation.

$$\Sigma = \mathbf{L} \mathbf{L}^T, \quad \mathbf{L} = \begin{pmatrix} \exp(L_0) & 0 & 0 \\ L_1 & \exp(L_2) & 0 \\ L_3 & L_4 & \exp(L_5) \end{pmatrix} \quad (2.4)$$

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[20] で事前学習した VGG16[21] のモジュールを採用する。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 [22]。最終出力層を除く全層は、活性化関数として ReLU 関数 [23] 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[24] を使用する。

## 2.2.4 損失関数

データセットの分布を学習し、出力の確率密度が最大になるようにネットワークパラメータを更新することで、DNN は平均と分散を出力することができる。推定が多変量正規分布に従うと仮定すると、推論で平均  $\hat{\mu}_\iota$ 、共分散行列  $\Sigma_\iota$  が与えられたとき、ラベルデータ  $\hat{\mathbf{g}}_\iota$  に対する確率密度は以下のように計算される。

$$p(\hat{\mathbf{g}}_\iota | \hat{\mu}_\iota, \Sigma_\iota) = \frac{\exp(-\frac{1}{2}(\hat{\mathbf{g}}_\iota - \hat{\mu}_\iota)^T \Sigma_\iota^{-1} (\hat{\mathbf{g}}_\iota - \hat{\mu}_\iota))}{\sqrt{(2\pi)^d |\Sigma_\iota|}}, \quad d = \text{rank}(\Sigma) \quad (2.5)$$

ここで、 $d$  は変数の次元 (i.e. 提案方法では  $d = 3$ ) である。ネットワークに確率モデルを学習させるために、 $p(\hat{\mathbf{g}}_\iota | \hat{\mu}_\iota, \Sigma_\iota)$  を最大化するように訓練する。データセット  $D^{\text{label}} =$

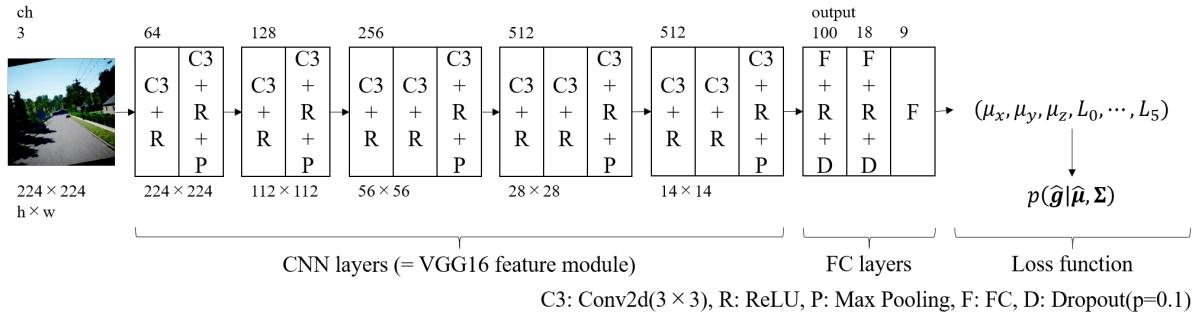


図 2.7: Proposed network architecture.

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.2.3, 2.4, respectively. Log-probability of multivariate normal distribution is used as a loss function of this model.

$\{\hat{g}_0, \dots, \hat{g}_\ell, \dots, \hat{g}_{\#D}\}$  全体に対する確率密度  $p_{\text{total}}$  は、以下のように乗算して計算される。

$$p_{\text{total}} = \prod_{\ell=0}^{\#D} p(\hat{g}_\ell | \hat{\mu}_\ell, \Sigma_\ell) \quad (2.6)$$

ここで、 $\#D$  はデータセットのサンプル数を表す。自然対数は単調増加関数なので、自然対数を取ることで、以下のように単純化することができる。

$$p_{\text{total}}^{\log} = \sum_{\ell=0}^{\#D} \ln p(\hat{g}_\ell | \hat{\mu}_\ell, \Sigma_\ell) \quad (2.7)$$

ここで、 $p_{\text{total}}^{\log}$  は対数尤度の合計を表す。これにより、乗算で値が小さくなりすぎるのを避けること、計算コストを下げることができる。さらに、 $p_{\text{total}}^{\log}$  を最大化することは、 $-p_{\text{total}}^{\log}$  を最小化することと同値であるため、提案手法の損失関数  $l_{(\Theta)}$  は以下のように定義される。

$$l_{(\Theta)} = -p_{\text{total}}^{\log} \quad (2.8)$$

ここで、 $\Theta$  はネットワークの重みパラメータを表す。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  $\Theta$  を更新する。

## 2.2.5 最適化

パラメータ  $\Theta$  の最適化には、Adam (adaptative moment estimation) [25] 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  $lr_{\text{CNN}} = 1 \times 10^{-5}$ ,  $lr_{\text{FC}} = 1 \times 10^{-4}$  とする。ここで、 $lr_{\text{CNN}}$  は CNN 層に適用される学習率、 $lr_{\text{FC}}$  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  $lr_{\text{CNN}} = 1 \times 10^{-6}$ ,  $lr_{\text{FC}} = 1 \times 10^{-5}$  である。

### 2.2.6 不確かさの表現

本研究では、以下の  $\eta$  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。

$$\eta = \sqrt{\sigma_x^2} \times \sqrt{\sigma_y^2} \times \sqrt{\sigma_z^2}, \quad \Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z^2 \end{pmatrix} \quad (2.9)$$

## 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 [11] 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。

### 2.3.1 水平な法線の抽出

LiDAR で取得する点群を  $\mathbf{C}$  とする。

$$\mathbf{C} = \{\mathbf{c}_0, \dots, \mathbf{c}_i, \dots, \mathbf{c}_{\#\mathbf{C}}\}, \quad \mathbf{c}_i = \begin{pmatrix} c_{i,x} & c_{i,y} & c_{i,z} \end{pmatrix} \quad (2.10)$$

ここで、 $\mathbf{c}_i$  はロボット座標系における各点の座標、 $\#\mathbf{C}$  は点の数を表す。各クエリ点の近傍点を k-dimensional tree[26] 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。

$$r^i = \alpha |\mathbf{c}_i| \quad (2.11)$$

$$\mathbf{B}^i = \{\dots, \mathbf{c}_k, \dots, \mathbf{c}_{\#\mathbf{B}^i}\} \quad (2.12)$$

ここで、 $\alpha$  は探索半径の比例定数、 $\mathbf{B}^i$  はクエリ点  $\mathbf{c}_i$  の近傍点群を表す。主成分分析 (PCA)[27] を各近傍点群に適用し、法線群  $\mathbf{N}$  を生成する。

$$\mathbf{N} = \{\mathbf{n}_0, \dots, \mathbf{n}_i, \dots, \mathbf{n}_{\#\mathbf{C}}\}, \quad \mathbf{n}_i = \begin{pmatrix} n_{i,a} & n_{i,b} & n_{i,c} & n_{i,d} \end{pmatrix} \quad (2.13)$$

ここで、 $\mathbf{n}_i$ はロボット座標系における法線の法線成分を表す (i.e.  $n_{i,a}x + n_{i,b}y + n_{i,c}z + n_{i,d} = 0$ )。平面度の高い鉛直平面の法線を以下の条件で選定する。ここで、選定された法線群を  $\check{\mathbf{N}}$  ( $\in \mathbf{N}$ ) とする。

$$\check{\mathbf{N}} = \{\dots, \mathbf{n}_{\check{i}}, \dots, \mathbf{n}_{\#\check{\mathbf{N}}}\} \quad (2.14)$$

- 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。

$$e^{\check{i}} = \sum_{k=0}^{\#B^{\check{i}}} \frac{|n_{i,a}c_{k,x} + n_{i,b}c_{k,y} + n_{i,c}c_{k,z} + n_{i,d}|}{|\mathbf{n}_{\check{i}}|} < \text{TH}_e \quad (2.15)$$

ここで、 $e^{\check{i}}$  は法線  $\mathbf{n}_{\check{i}}$  におけるフィッティング誤差、 $\text{TH}_e$  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。

$$\#B^{\check{i}} > \text{TH}_{\#B} \quad (2.16)$$

ここで、 $\text{TH}_{\#B}$  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、 $\#B^{\check{i}}$  が小さすぎると、 $e^{\check{i}}$  が小さくなりやすいため、この閾値は  $e^{\check{i}}$  を有効なものにするためのものもある。

- 法線  $\mathbf{n}_{\check{i}}$  が水平に十分近い。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。

$$\beta^{\check{i}} = \left| \cos^{-1} \frac{\mathbf{n}_{\check{i}} \cdot \boldsymbol{\mu}_{t-1}}{|\mathbf{n}_{\check{i}}| |\boldsymbol{\mu}_{t-1}|} - \frac{\pi}{2} \right| < \text{TH}_{\beta} \quad (2.17)$$

ここで、 $\beta^{\check{i}}$  は法線  $\mathbf{n}_{\check{i}}$  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、 $\text{TH}_{\beta}$  はその角度の閾値を表す。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

選定された法線群  $\check{\mathbf{N}}$  の全ての始点を、原点に移動させて点群  $\mathbf{S}$  を生成する。清水ら [28] はこの点群を ‘depth-Gaussian sphere’ と呼んでいる (cf. 我々の先行研究 [11] では Gaussian sphere[29] が用いられている)。図 2.8 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。

$$\mathbf{S} = \{\dots, \mathbf{s}_{\check{i}}, \dots, \mathbf{s}_{\#\check{\mathbf{N}}}\}, \quad \mathbf{s}_{\check{i}} = -n_{i,d} \begin{pmatrix} n_{i,a} & n_{i,b} & n_{i,c} \end{pmatrix} \quad (2.18)$$

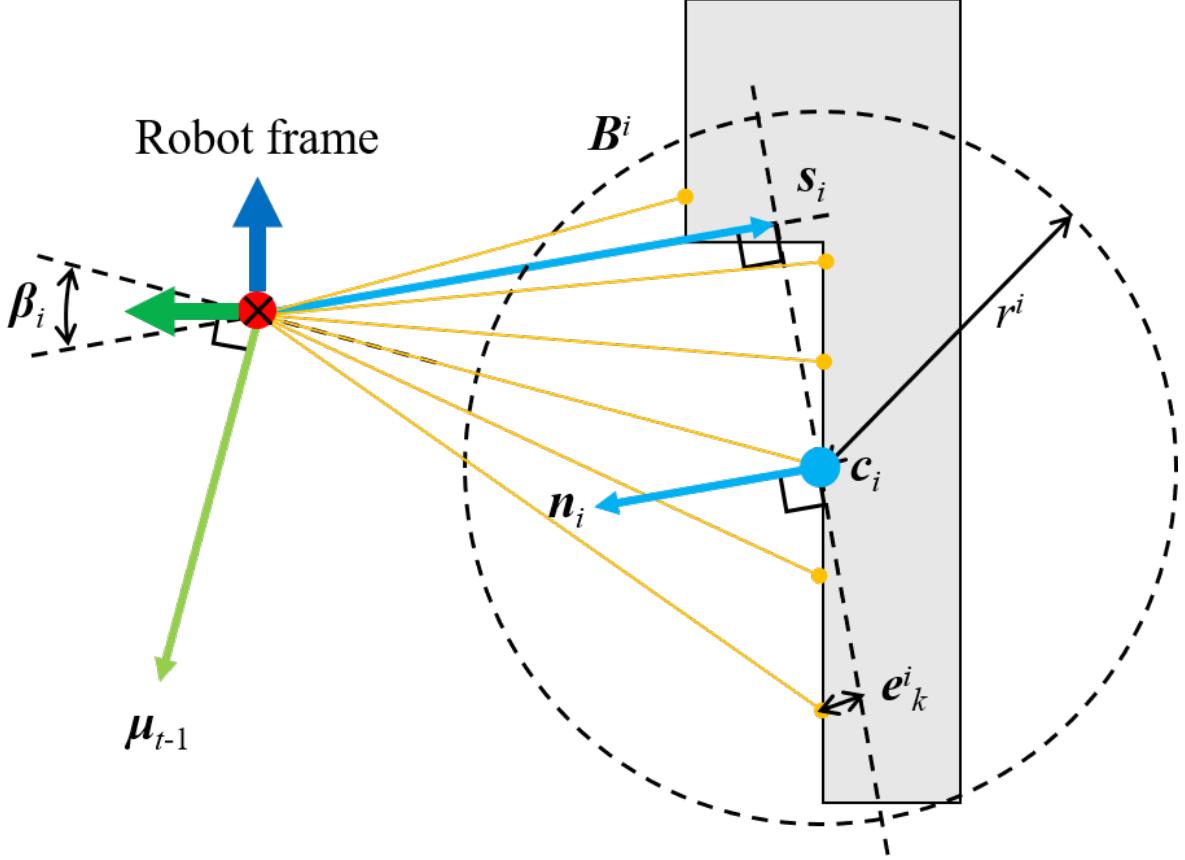


図 2.8: Depth-Gaussian sphere.

Noramls of vertical flat planes are extracted form LiDAR point cloud, and they are converted to ‘depth-Gaussian sphere’  $S = \{\dots, s_i, \dots\}$ .

### 2.3.3 depth-Gaussian sphere のクラスタリング

点群 (depth-Gaussian sphere)  $S$  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 [11] 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,  $W$  はクラスタの集合,  $w_j$  は環境内の各平面 (壁) を表す.

$$W = \{w_0, \dots, w_j, \dots, w_{\#W}\}, \quad w_j = \{s_0^j, \dots, s_l^j, \dots, s_{\#w_j}^j\} \quad (2.19)$$

式 2.20 を満たすとき, 式 2.21 のように点  $s_i$  はクラスタ  $w_j$  に分類される.

$$\begin{aligned} \gamma^j &= \min\{\gamma_+^j, \gamma_-^j\} < \text{TH}_\gamma \\ \gamma_+^j &= \cos^{-1} \frac{s_i \cdot \sum_{l=0}^{\#w_j} s_l^j}{|s_i| \sum_{l=0}^{\#w_j} s_l^j}, \quad \gamma_-^j = \cos^{-1} \frac{-s_i \cdot \sum_{l=0}^{\#w_j} s_l^j}{|-s_i| \sum_{l=0}^{\#w_j} s_l^j} \end{aligned} \quad (2.20)$$

$$w_j+ = s_i \quad (2.21)$$

式 2.20 を満たさないときは, 式 2.22 のように点  $s_i$  は新しいクラスタとして登録される.

$$W+ = \{s_i\} \quad (2.22)$$

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は大きくなる。角度  $\gamma^j$  の例を図1に示す。 $\gamma_+^j$  だけでなく  $\gamma_-^j$  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 2.23 で定義されるクラスター  $\check{W}$  が重力ベクトルの推定に使用されることになる。

$$\check{W} = \{\dots, w_j, \dots, w_{\#\check{W}}\}, \quad \#w_j > TH_{\#w} \quad (2.23)$$

ここで、 $TH_{\#w}$  はメンバ数に対する閾値である。

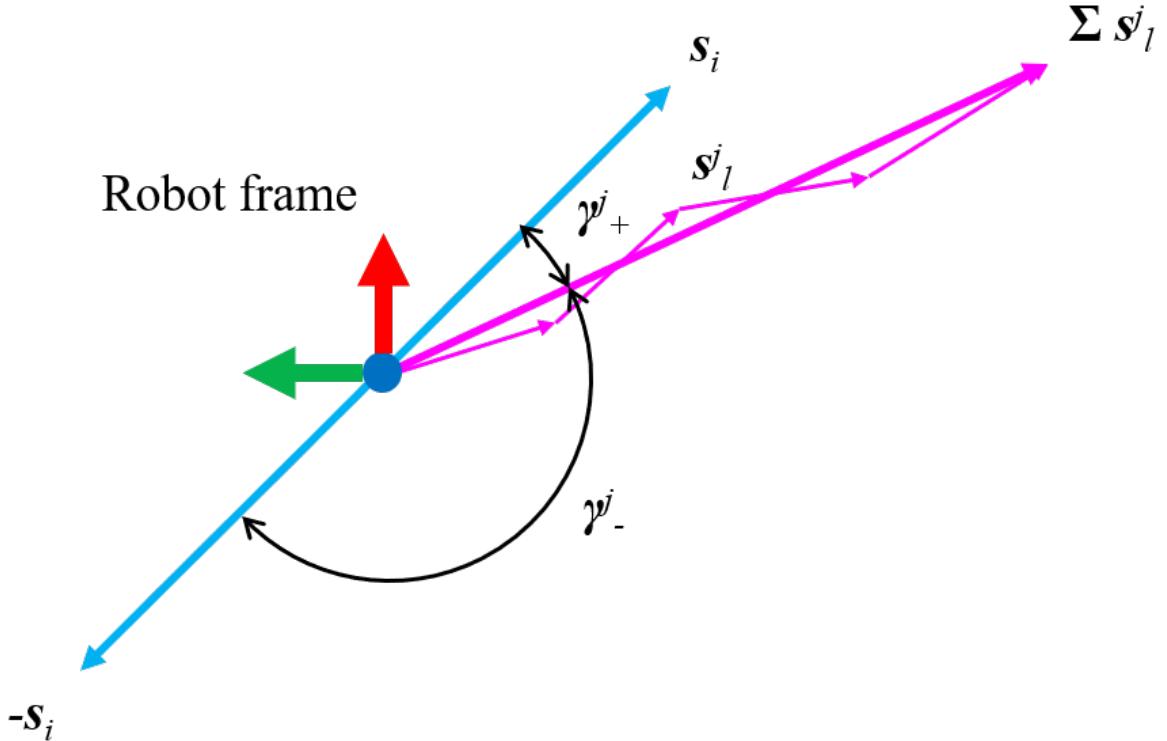


図 2.9: Angle-based clustering.

Points of depth-Gaussian sphere are clustered based on the angle  $\gamma_+^j$  and  $\gamma_-^j$ . 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  $\mu$  をクラスタ  $\check{W}$  を用いて推定する。推定は、後述するように、クラスタの数  $\#\check{W}$  に応じて、以下の各方法で計算される。

- $\#\check{W} > 1$  の場合、クラスターの外積が重力ベクトル  $\mu$  として推定される。図 2.10a

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。

$$\boldsymbol{\mu} = \sum_{j', j'' (j' \neq j'')}^{\#\check{W}} \left( \sum_{l=0}^{\#w_{j'}} \mathbf{s}_l^{j'} \times \sum_{l=0}^{\#w_{j''}} \mathbf{s}_l^{j''} \right) \quad (2.24)$$

前ステップの推定値  $\boldsymbol{\mu}_{t-1}$  と外積  $\sum \mathbf{s}_l^{j'} \times \sum \mathbf{s}_l^{j''}$  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。

- $\#\check{W} = 1$  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 2.10b に計算例を示す。

$$\boldsymbol{\mu} = \boldsymbol{\mu}_{t-1} - \left( \boldsymbol{\mu}_{t-1} \cdot \sum_{l=0}^{\#w_j} \mathbf{s}_l^j \right) \sum_{l=0}^{\#w_j} \mathbf{s}_l^j \quad (2.25)$$

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

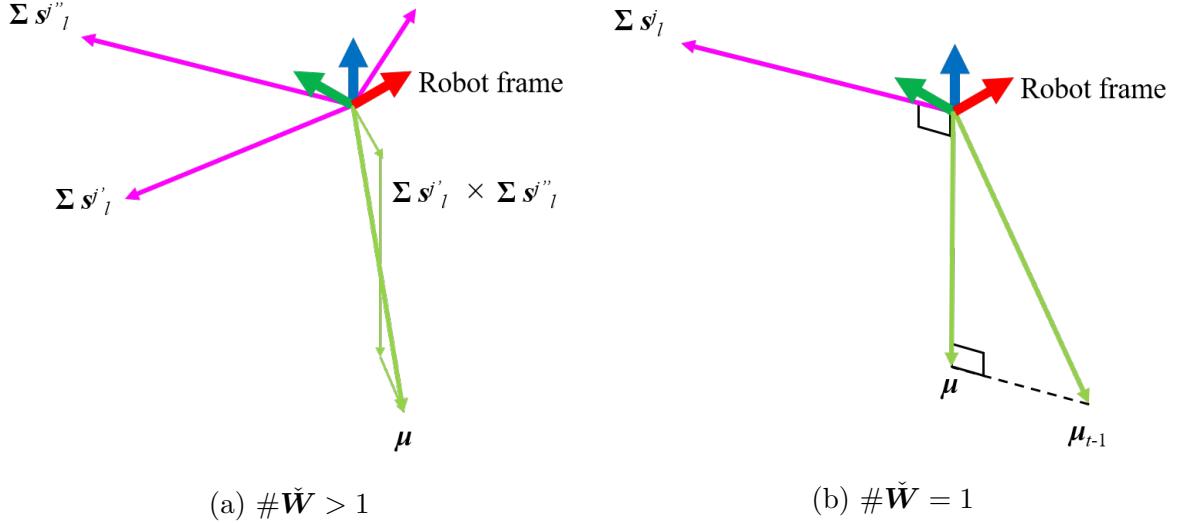


図 2.10: Gravity vector estimation.

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 2.1 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  $\mathbf{x}$  は、ロボットのロール  $\phi$  とピッチ  $\theta$  で構成されている。

$$\mathbf{x} = \begin{pmatrix} \phi & \theta \end{pmatrix}^T \quad (2.26)$$

状態ベクトル  $\mathbf{x}$  と共に分散行列  $\mathbf{P}$  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、 $S_\phi$ ,  $C_\phi$ ,  $T_\phi$  はそれぞれ  $\sin \phi$ ,  $\cos \phi$ ,  $\tan \phi$  の略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  $\mathbf{x}$  と共に分散行列  $\mathbf{P}$  は、それぞれ以下のように計算される。

$$\bar{\mathbf{x}}_t = f_{(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})} = \mathbf{x}_{t-1} + \mathbf{Rot}_{(\mathbf{x}_{t-1})}^{\text{rpy}} \mathbf{u}_{t-1}$$

$$\mathbf{u}_{t-1} = \boldsymbol{\omega}_{t-1} \Delta t = \begin{pmatrix} \omega_{x_{t-1}} \Delta t \\ \omega_{y_{t-1}} \Delta t \\ \omega_{z_{t-1}} \Delta t \end{pmatrix}, \quad \mathbf{Rot}_{(\mathbf{x}_{t-1})}^{\text{rpy}} = \begin{pmatrix} 1 & S_{\phi_{t-1}} T_{\theta_{t-1}} & C_{\phi_{t-1}} T_{\theta_{t-1}} \\ 0 & C_{\phi_{t-1}} & -S_{\phi_{t-1}} \end{pmatrix} \quad (2.27)$$

ここで、 $f$  は状態遷移モデル、 $\mathbf{u}$  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、 $\mathbf{Rot}^{\text{rpy}}$  は角速度の回転行列を表す。

$$\bar{\mathbf{P}}_t = \mathbf{J}_{f_{t-1}} \mathbf{P}_{t-1} \mathbf{J}_{f_{t-1}}^T + \mathbf{Q}_{t-1}, \quad \mathbf{J}_{f_{t-1}} = \frac{\partial f}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{t-1}, \mathbf{u}_{t-1}} \quad (2.28)$$

ここで、 $\mathbf{J}_f$  は  $f$  のヤコビアン、 $\mathbf{Q}$  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  $\eta$  が大きい場合、DNNの出力は棄却される。 $\eta$  の判定には閾値  $\text{TH}_\eta$  が設定されており、 $\eta < \text{TH}_\eta$  を満たす推論のみがEKFで観測される。観測ベクトルを  $z$  とする。

$$z = \hat{\mu}_{\text{camera}} \quad (2.29)$$

ここで、 $\hat{\mu}_{\text{camera}}$  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。

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

$$\mathbf{Rot}_{(\mathbf{x}_t)}^{\text{xyz}} = \begin{pmatrix} C_{\theta_t} C_{\psi_t} & S_{\phi_t} S_{\theta_t} C_{\psi_t} - C_{\phi_t} S_{\psi_t} & C_{\phi_t} S_{\theta_t} C_{\psi_t} + S_{\phi_t} S_{\psi_t} \\ C_{\theta_t} S_{\psi_t} & S_{\phi_t} S_{\theta_t} S_{\psi_t} + C_{\phi_t} C_{\psi_t} & C_{\phi_t} S_{\theta_t} S_{\psi_t} - S_{\phi_t} C_{\psi_t} \\ -S_{\theta_t} & S_{\phi_t} C_{\theta_t} & C_{\phi_t} C_{\theta_t} \end{pmatrix} \quad (2.30)$$

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

$$\mathbf{R} = \begin{pmatrix} \xi\sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \xi\sigma_y^2 & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \xi\sigma_z^2 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z^2 \end{pmatrix} \quad (2.31)$$

ここで,  $\xi$  は分散を調整するためのハイパーパラメータとする. 状態ベクトル  $\mathbf{x}$  と共分散行列  $\mathbf{P}$  は, それぞれ以下のように計算される.

$$\begin{aligned} \tilde{\mathbf{x}}_t &= \mathbf{x}_t + \mathbf{K}_t(\mathbf{z}_t - h_{(\mathbf{x}_t)}), & \tilde{\mathbf{P}}_t &= (\mathbf{I} - \mathbf{K}_t \mathbf{J}_{ht}) \mathbf{P}_t \\ \mathbf{J}_{ht} &= \left. \frac{\partial h}{\partial \mathbf{x}} \right|_{\mathbf{x}_t}, & \mathbf{K}_t &= \mathbf{P}_t \mathbf{J}_{ht}^T (\mathbf{J}_{ht} \mathbf{P}_t \mathbf{J}_{ht}^T + \mathbf{R}_t)^{-1} \end{aligned} \quad (2.32)$$

ここで,  $\mathbf{J}_h$  は  $h$  のヤコビアン,  $\mathbf{K}$  はゲイン行列,  $\mathbf{I}$  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  $\mathbf{z}$  を用いて, 式 2.30, 2.32 と同様に, 状態ベクトル  $\mathbf{x}$  と共分散行列  $\mathbf{P}$  をそれぞれ更新する.

$$\mathbf{z} = \frac{\boldsymbol{\mu}_{\text{lidar}}}{|\boldsymbol{\mu}_{\text{lidar}}|} \quad (2.33)$$

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.

# 第3章 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2.9の $\eta$ が推論の不確かさを表すと仮定して、閾値 $TH_\eta$ 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 $TH_\eta = \frac{1}{\#D} \sum_{\ell=0}^{\#D} \eta_\ell$ 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[16]に基づいて実装されている。ただし、関連研究[16]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。
- ‘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 $\sum_{\ell=0}^{\#D} g_\ell$ 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3.1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 3.1 にプロットした。回帰モデルは、MLE モデルよりもはるかに早く収束した。表 3.2 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。

表 3.1: Dataset list.

id#	Environment		#samples	Usage
1	Sim.	AirSim's 'Neighborhood'	10000	Training
2			1000	Test
3	Real	AreaI		Fine-tuning
4		AreaII (daytime)		Test
5		AreaII (nighttime)		Test

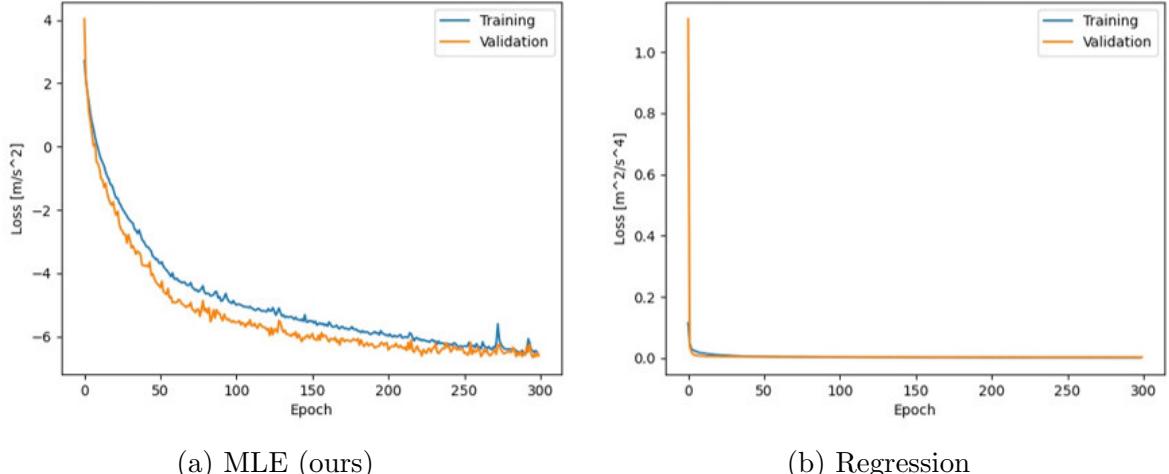


図 3.1: Loss plotting of training.

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.

表 3.2: Loss after 300 epochs of training.

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

### 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3.2にプロットした。表3.3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ている。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ている。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。

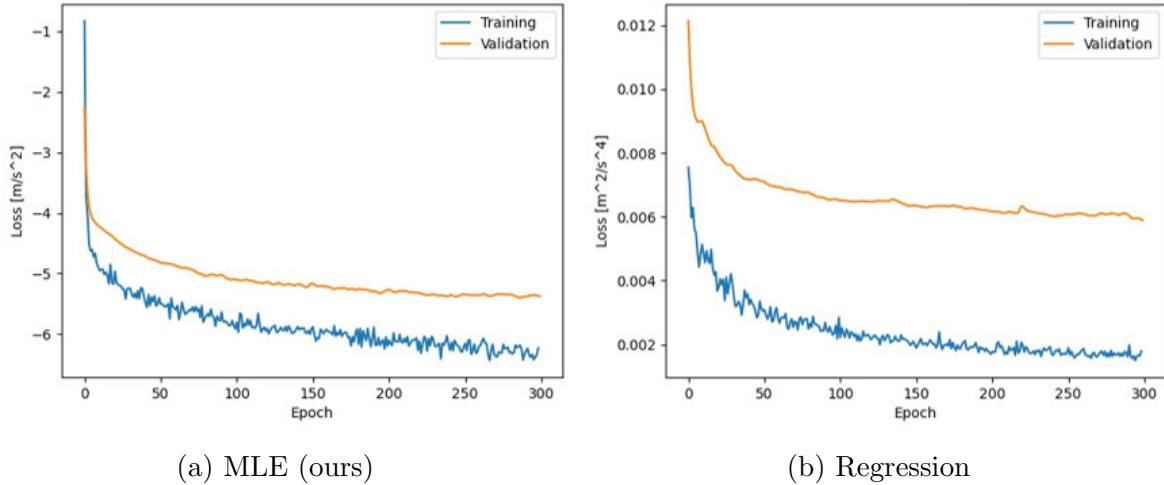


図3.2: Loss plotting of fine-tuning.

The fine-tuning made the loss values on the real data smaller.

表3.3: Loss after 300 epochs of fine-tuning.

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

### 3.1.4 静的姿勢推定

$\hat{\mu}$  ( $= \hat{\mu}_{\text{camera}}$ ) を用いて、重力座標系におけるロボット姿勢のロール  $\phi$ , ピッチ  $\theta$  が推定される。

$$\phi = \tan^{-1} \frac{\hat{\mu}_y}{\hat{\mu}_z}, \quad \theta = \tan^{-1} \frac{-\hat{\mu}_x}{\sqrt{\hat{\mu}_y^2 + \hat{\mu}_z^2}} \quad (3.1)$$

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 3.4 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、 $\eta < \text{TH}_\eta = \frac{1}{\#D} \sum_{i=0}^{\#D} \eta_i = 0.000120 \text{ m}^3/\text{s}^6$  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 3.5 に示す。実データセットでの推定値の MAE を表 3.6 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、 $\text{TH}_\eta$  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。これを確認するために、図 3.3 に、Dataset#2 のうち推論された分散値  $\eta$  が最も小さい 10 サンプルと、最も大きい 10 サンプルを示す。図 3.3b のサンプルは、明らかに、重力方向を推定するための風景情報が非常に少なく、誤差も大きい傾向が見られる。これに対し、提案ネットワークは、大きい  $\eta$  を出力することで不確かさを表現している。一方で、従来の回帰モデルではこれらを検出する方法がない。

‘before fine-tuning’ と ‘after fine-tuning’ を比較すると、実データを用いたファインチューニングにより誤差が小さくなった。ファインチューニングのためのサンプル数は多くはないが、十分小さい誤差で推定することができた。これは、大量なシミュレーションデータセットを用いた事前学習が有効であることを示唆している。

先行研究 [17] 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) に比べて大きくなかった。‘MLE w/ rejection (ours)’ の誤差が大きくならなかったのは、図 3.4 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  $\text{TH}_\eta$  で選択されたからだと考察できる。ただし、表 3.5 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。

## 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

表 3.4: MAE of static estimation on synthetic data.

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

表 3.5: Number of samples selected by MAE w/ rejection (ours).

#Selected samples (percentage [%])	Dataset#				
	1	2	3	4	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)

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。

### 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[5] 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、回帰ネットワーク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。

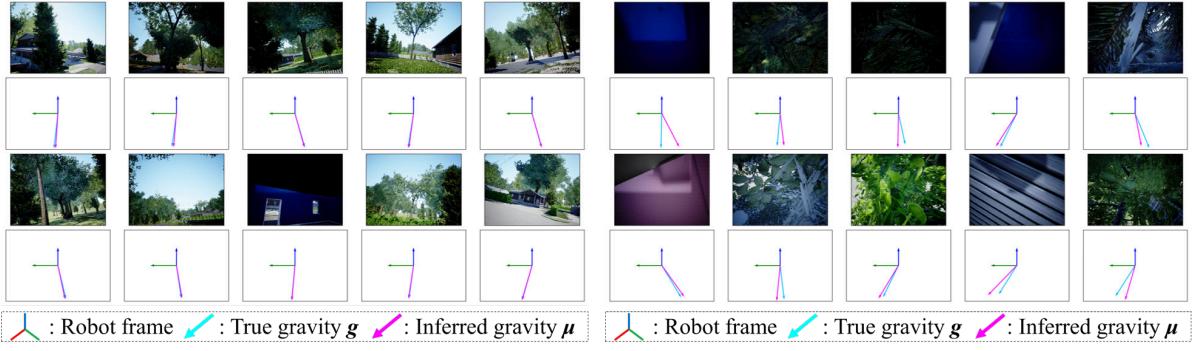
表 3.6: MAE of static estimation on real data.

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

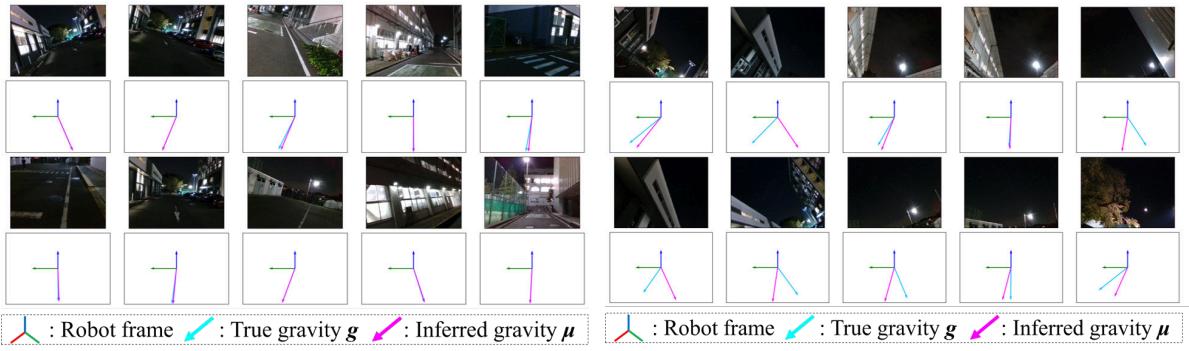
- ‘Gyro+MLE’ は、角速度センサで計測する角速度と、提案 DNN (図 2.7) の推論を統合する EKF ベースの推定手法を表す。推論される分散値  $\eta$  が大きい場合、推論は統合されない。ハイパーパラメータは、 $TH_\eta = 1.2 \times 10^{-4} \text{ m}^3/\text{s}^6$ ,  $\xi = 1 \times 10^3$  と設定される。これらは、3.1 節の検証結果や経験を基に決められた。
- ‘Gyro+DGSphere’ は、角速度センサで計測する角速度と、LiDAR を用いた手法 (2.3 節) で得られる重力ベクトルを統合する 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 3.7 に示す。

### 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<sup>2</sup>, 標準偏差 0.5 rad/s, 0.5 m/s<sup>2</sup> の正規分布に沿ってランダムに付与された。飛行コースを図 3.5a に示す。

(a) Top 10 smallest variance  $\eta$ (b) Top 10 largest variance  $\eta$ 図 3.3: Synthetic samples (Dataset#2) sorted in  $\eta$  values.

The DNN tends to output large error and variance when the image does not have enough landscape information.

(a) Top 10 smallest variance  $\eta$ (b) Top 10 largest variance  $\eta$ 図 3.4: Real samples (Dataset#5) sorted in  $\eta$  values.

The DNN tends to output large error and variance when the image is dark.

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。

### 3.2.3 実験結果

実験中に推定された姿勢を図 3.6 にプロットした。表 3.8 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かった。‘Gyro’ は累積誤差が大きくなかった。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。

表 3.7: Hyperparameters.

$\alpha$	0.09
$\text{TH}_e$	0.05 m
$\text{TH}_{\#B}$	10
$\text{TH}_\beta$	15 deg
$\text{TH}_\gamma$	5 deg
$\text{TH}_{\#w}$	20
$\text{TH}_\eta$	$1.2 \times 10^{-4} \text{ m}^3/\text{s}^6$
$\xi$	$1 \times 10^3$

‘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、 $\eta$ 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  $\text{TH}_\eta$ 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  $\eta$  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 節) は約 51600 回、カメラを用いた更新プロセス (2.4.2 節) は約 3100 回、LiDAR を用いた更新プロセス (2.4.3 節) 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。結果を見ることで、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。提案手法は、将来的には、IMU で計測される加速度や SLAM などの他の観測を統合することも可能である。一方、本実験では、評価をより簡単にするために、角速度と DNN の出力、点群処理の出力のみを統合している。

表 3.8: MAE of dynamic estimation in simulator.

	Roll [deg]	Pitch [deg]
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)	<b>2.772</b>	<b>2.141</b>



(a) Simulator

(b) Real world

図 3.5: Driving course.

The drone flew for about 9 minutes in (a). The sensors were carried for about 5 minutes in (b).

### 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 2.4）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。

#### 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 3.9 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。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 [30]、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。

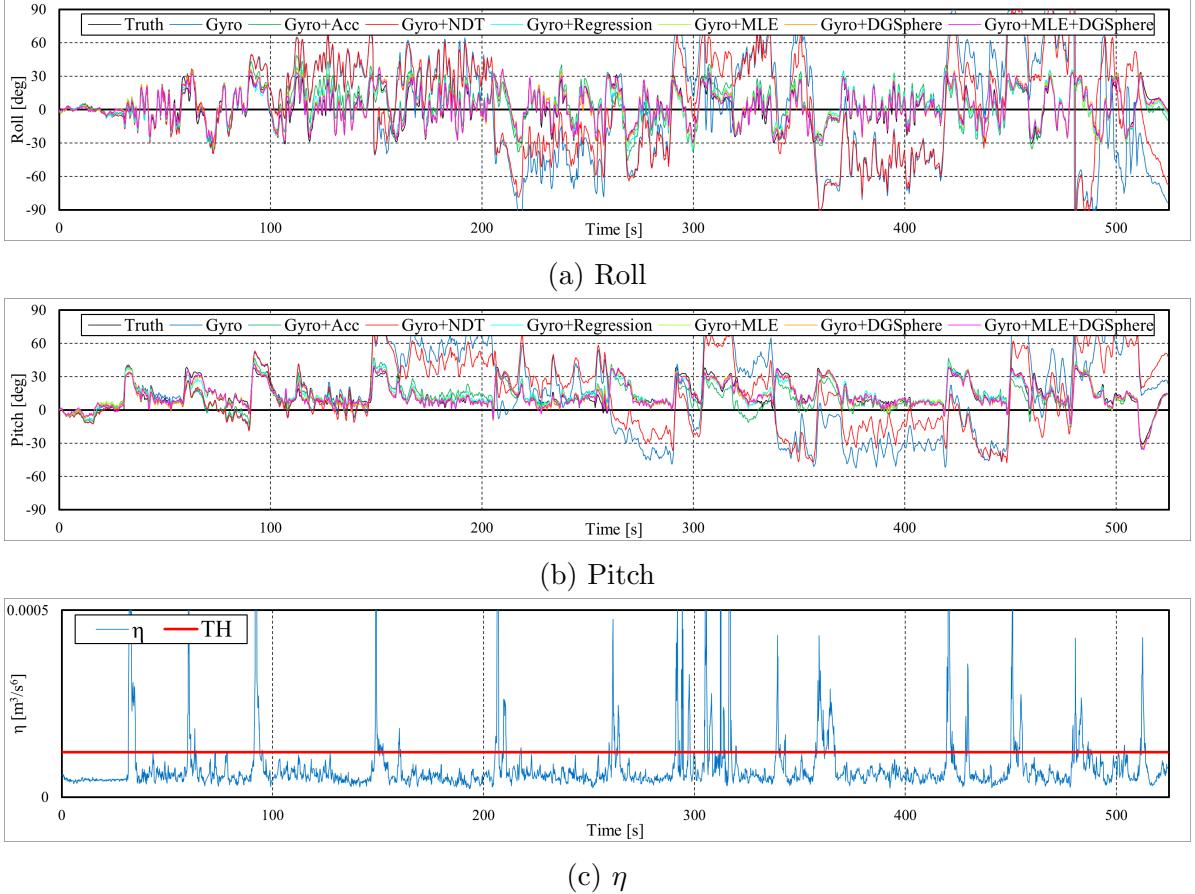


図 3.6: Real-time plotting in ‘Neighborhood’.

The graphs show the estimated attitude and the inferred variance in the synthetic flight. ‘MLE’ tended to output large variance  $\eta$  when the angles exceed the trained range [-30 deg, 30 deg].

AreaII (図 3.5b) 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. 実験の開始時と終了時に, 図 3.7 のように, 平らな床面にセンサを置き, そのときの真値を  $\phi_{gt} = 0$  deg,  $\theta_{gt} = 0$  deg と仮定した. この評価方法は, 関連研究 [16] に基づくものである.

表 3.10 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.

### 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) にはさまざまな閾値の設定が必要である.

表 3.9: MAE of dynamic estimation in mocap area.

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

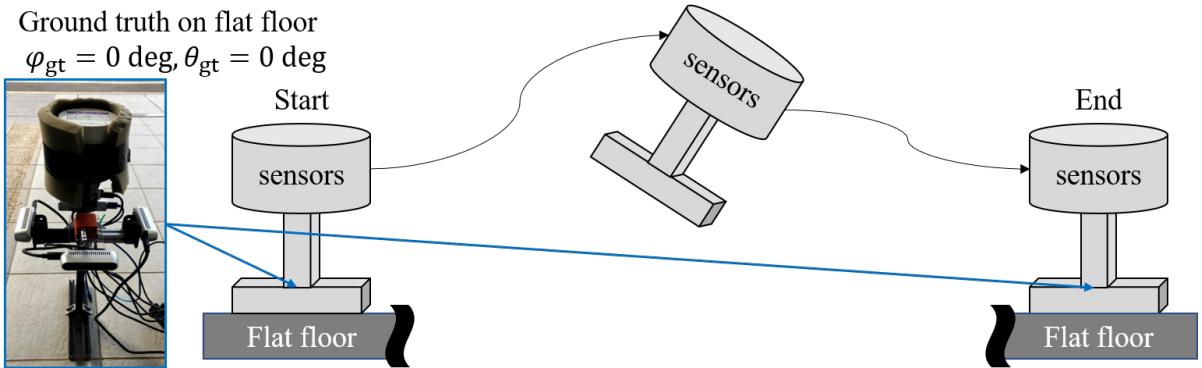


図 3.7: Dynamic experiment.

The ground truth on the flat floor is assumed to be  $\varphi_{gt} = 0 \text{ deg}, \theta_{gt} = 0 \text{ deg}$ .

表 3.10: Error of estimated attitude at last pose in outdoor.

	Roll [deg]	Pitch [deg]
Gyro+MLE+DGSphere (ours)	<b>+0.515</b>	<b>+2.110</b>
Gyro	+5.268	-5.047

## 第4章 結論

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 の統合時にプロセス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。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すことが示された。

本論文ではセンサースイートを手で持って移動したが、実際の移動ロボットでの検証を今後の課題として設定する。特に、ロボットの上体を傾けるなどの姿勢制御のために、提案した推定法を利用できるか検証する必要がある。

# 謝辞

本研究を進めるにあたり、指導教員の黒田洋司教授から技術的な助言、原稿の添削を受けた。さらに、松田匠未助教、ロボット工学研究室の学生の皆様にも多くの助言を頂いた。また、本研究は明治大学自律型ロボット研究クラスターの下で実施された。ここに篤く御礼申し上げる。

明治大学理工学研究科  
機械工学専攻  
ロボット工学研究室修士 2 年  
2021 年 2 月 尾崎亮太

# 参考文献

- [1] D. Titterton, J.L. Weston, and J. Weston. *Strapdown Inertial Navigation Technology*, Vol. 17. IET, 2004.
- [2] J. Vaganay, M. J. Aldon, and A. Fournier. Mobile robot attitude estimation by fusion of inertial data. In *Proceedings of 1993 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 277–282, 1993.
- [3] S. Thrun, W. Burgard, and D. Fox. In *Probabilistic robotics*, pp. 309–336. The MIT Press, 2005.
- [4] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings of Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152, 2001.
- [5] P. Biber and W. Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.
- [6] J. Engel, J. Stueckler, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 834–849, 2014.
- [7] 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.
- [8] 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, pp. 312–328, 2007.
- [9] P. Kim, B. Coltin, and H. J. Kim. Linear rgb-d slam for planar environments. In *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 333–348, 2018.

- [10] M. Hwangbo and T. Kanade. Visual-inertial uav attitude estimation using urban scene regularities. In *Proceedings of 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2451–2458, 2011.
- [11] 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, pp. 19–00065, 2019.
- [12] 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, p. 3777, 2019.
- [13] 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.
- [14] 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. In *Proceedings of 2019 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5672–5678, 2019.
- [15] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *Field and Service Robotics*, pp. 621–635, 2018.
- [16] G. Ellingson, D. Wingate, and T. McLain. Deep visual gravity vector detection for unmanned aircraft attitude estimation. In *Proceedings of 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5557–5563, 2017.
- [17] R. Ozaki and Y. Kuroda. Dnn-based self-attitude estimation by learning landscape information. In *Proceedings of 2021 IEEE/SICE International Symposium on System Integration (SII2021)*, 2021.
- [18] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, Vol. 82, pp. 35–45, 1960.
- [19] S. J. Julier and J. K. Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal Processing, Sensor Fusion, and Target Recognition VI*, Vol. 3068, pp. 182–193, 1997.
- [20] J. Deng, W. Dong, R. Socher, Kai Li L. Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

- [21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv preprint*, p. arXiv:1409.1556, 2014.
- [22] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 22, No. 10, pp. 1345–1359, 2010.
- [23] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of ICML 2010*, pp. 807–814, 2010.
- [24] 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.
- [25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*, 2015.
- [26] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, Vol. 18, No. 9, pp. 509–517, 1975.
- [27] M. Pauly, M. Gross, and LP. Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the Conference on Visualization*, 2002.
- [28] 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, pp. 453–458, 2014.
- [29] B.K.P. Horn. Extended gaussian images. *Procceeding of the IEEE*, Vol. 72, No. 12, pp. 1671–1686, 1984.
- [30] B. Suwandi, T. Kitasuka, and M. Aritsugi. Vehicle vibration error compensation on imu-accelerometer sensor using adaptive filter and low-pass filter approaches. *Journal of Information Processing*, Vol. 27, pp. 33–40, 2019.

## 付 錄A 公開データ

- DNN の学習に用いられたデータセット.

[https://github.com/ozakiryota/dataset\\_image\\_to\\_gravity](https://github.com/ozakiryota/dataset_image_to_gravity)

- DNN の学習に用いられたソースコード. これは, Python, PyTorch API を用いて実装されている.

[https://github.com/ozakiryota/image\\_to\\_gravity](https://github.com/ozakiryota/image_to_gravity)

- DNN の推論に用いられたソースコード. これは, Python, PyTorch API, ROS API を用いて実装されている.

[https://github.com/ozakiryota/dnn\\_attitude\\_estimation](https://github.com/ozakiryota/dnn_attitude_estimation)

- LiDAR 点群の処理および EKF を実装したソースコード. これは, C++, ROS API を用いて実装されている.

[https://github.com/ozakiryota/attitude\\_estimation\\_walls](https://github.com/ozakiryota/attitude_estimation_walls)

# 付 錄B 発表業績一覧

## 学術雑誌

- 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, pp.19-00065, 2019/7/25.
- Ryota Ozaki and Yoji Kuroda, "EKF-based self-attitude estimation with DNN learning landscape information," ROBOMECH Journal, Vol.?, No.?, pp.?-, 2021/?/? (in press).
- Ryota Ozaki and Yoji Kuroda, "EKF-based real-time self-attitude estimation with camera DNN learning landscape regularities," IEEE Robotics and Automation Letters (RA-L), Vol.?, No.?, pp.?-, 2021/?/? (in press).

## 講演会

### 査読あり

- 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姿勢推定」, 第 24 回ロボティクスシンポジア, pp.120-121, 2019/3/15.
- Ryota Ozaki and Yoji Kuroda, "6-DoF EKF SLAM with Global Planar Features in Artificial Environments," Proc. of 2020 IEEE/SICE International Symposium on System Integrations (SII 2020), pp.531-535, 2020/1/14.
- 尾崎亮太, 黒田洋司, 「人工環境における平面特徴量を用いたランドマーク SLAM」, 第 25 回ロボティクスシンポジア, pp.316-317, 2020/3/15.
- Ryota Ozaki and Yoji Kuroda, "DNN-Based Self-Attitude Estimation by Learning Landscape Information," Proc. of 2021 IEEE/SICE International Symposium on System Integrations (SII 2021), pp.733-738, 2021/1/13.
- 尾崎亮太, 黒田洋司, 「風景知識を学習するカメラ-LiDAR DNN による自己姿勢推定」, 第 26 回ロボティクスシンポジア, pp.?-, 2021/3/16 or 17 (発表予定) .

## 査読なし

- 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 6DoF 位置姿勢推定」, 関東学生会第 58 回学生員卒業研究発表講演会, 2019/3/18.
- 尾崎亮太, 黒田洋司, 「鉛直壁面を用いた移動ロボットのための自己姿勢推定」, 日本機械学会ロボティクス・メカトロニクス講演会 2019, 2019/6/6.
- 尾崎亮太, 黒田洋司, 「人工環境の平面をランドマークとして用いる EKF-SLAM」, 第 20 回計測自動制御学会システムインテグレーション部門講演会 (SI2019), pp.165-166, 2019/12/12.

## その他

- 恩田和弥, 大石朋孝, 有馬純平, 尾崎亮太, 隼田駿大, 黒田洋司, 「Edge-node Map 及び交差点形状マッチングを用いたナビゲーションシステムの開発」, つくばチャレンジシンポジウム, pp.101-106, 2019/1/14.
- 尾崎亮太, 「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」, 2018 年度 卒業論文, 2019/2/2.
- Ryota Ozaki and Yoji Kuroda, “6-DoF EKF SLAM with global planar features in artificial environments,” The Fourteenth International Symposium on Mechanics, Aerospace and Informatics Engineering 2019 (ISMAI-14), pp. 215-218, 2019/9/4.



## 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

尾崎 亮太 \*1, 黒田 洋司 \*2

## Real-time 6DoF localization with relative poses to walls of buildings

Ryota OZAKI\*1 and Yoji KURODA\*2

\*1,\*2 Department of Mechanical Engineering, School of Science and Technology, Meiji University  
1-1-1 Higashimita, Kawasaki Tama-ku, Kanagawa 214-8571, Japan

Received: 13 February 2019; Revised: 6 May 2019; Accepted: 5 June 2019

## Abstract

This paper presents a real-time 6DoF localization method which corrects accumulative error by estimating relative poses to building walls for mobile robots in urban areas. This method exploits a fact that most of all artificial walls are built vertically. It estimates poses by not only an inertial sensor but also real-time SLAM and observations of normals from point-cloud of artificial walls for estimating absolute poses in the gravity coordinate system. Those three types of poses which are estimated by each way are combined by extended Kalman filter. To evaluate the proposed method, outdoor experiments with an actual robot were performed. They show the method keeps correcting accumulative error while the robot moves. The other experiments show how the method suppresses influence of non-vertical planes.

**Keywords :** Pose estimation, 6DoF localization, Mobile robot, Artificial environment, Gauss map, Extended Kalman filter

## 1. 緒 言

近年、屋内や市街地のような整備された人工環境において活動する自律移動ロボットへの需要が高まっている。また、次世代の移動ロボットは、その適用範囲を広げるため、高い機動性を持つことが要求される。高い機動性を実現するための手段の1つとして姿勢制御があり、実際に開発されている(Kwon et al., 2015)。移動ロボットの姿勢制御を行うには、走行中における時々刻々の3次元空間におけるロボットの自己姿勢を推定し把握し続ける必要がある。ここで、本論文では、重力方向ベクトルに対するロボットの傾きを姿勢と呼ぶこととする。なお、この姿勢をワールド座標系で表現する場合は、重力ベクトルに対するロール、ピッチに加えて、ヨーも必要となる。

移動体の姿勢は、一般に、ジャイロスコープや加速度センサを使って推定することが多い。地面上を走行する移動ロボットの場合、ロボット自身の加速度や地面からの振動が推定精度に大きな影響を与えるため、さらに外界センサを用いたSLAM (Simultaneous Localization and Mapping) (Thrun et al., 2005)との組み合わせが提案されている(Stumberg et al., 2018)。姿勢を推定するためのSLAMとして、ICPスキャンマッチング(Rusinkiewicz and Levoy, 2001)やNDTスキャンマッチング(Biber and Straßer, 2003)(Magnusson et al., 2007)を用いたSLAMが代表的である。これらのスキャンマッチングでは、2つの3次元点群に対して共通領域を基に点群の位置合わせを行うことで相対位置姿勢を推定する。また、Zhangら(Zhang and Singh, 2014)はこの位置合わせにおいて、平面やエッジの特徴量を活用している。一方、清水ら(清水, 黒田, 2014)は、点群そのままではなくクラスタリングした法

No.19-00065 [DOI:10.1299/transjsme.19-00065], J-STAGE Advance Publication date : 13 June, 2019

\*1 学生員, 明治大学 理工学部機械工学科 (〒214-8571 神奈川県川崎市多摩区東三田 1-1-1)

\*2 明治大学 理工学部機械工学科

E-mail of corresponding author: ee53031@meiji.ac.jp

線をガウス球に投影し、支配的な法線（主平面）で位置合わせを行っている。また、カメラやRGB-Dカメラを用いたSLAMによる相対姿勢推定も提案されている(Engel et al., 2014)(Kerl et al., 2013)(Engel et al., 2018)。これらの手法は、初期姿勢に対する相対変化を推定するため、蓄積誤差を補正することが難しい。この蓄積誤差を補正する方法として、事前情報と時々刻々のセンサ情報を対応付けする手法があり、特に事前環境地図を用いる手法が多く提案されている(Quddus et al., 2007)。また、人工環境での特徴を利用した手法も以下のように提案されている。S. Ramalingamら(Ramalingam et al., 2010)は、街の3次元モデルと、全点球カメラの画像をマッチングを行う。このマッチングでは、空を見上げた時の空と建物の境界線(skylines)を用いている。後藤ら(後藤他, 2017)は同様にモデルと全点球カメラの対応付けを提案しており、天井から床に至る線分および天井と壁、床と壁との線分など、人工物環境に多数存在する線分をマッチングの特徴量として用いて6自由度位置姿勢推定を行っている。Alberto Y. Hataら(Hata et al., 2014)は自動車のための自己位置推定を提案しており、縁石の情報を含んだ地図と、センサで観測した縁石とのマッチングを行っている。このような環境地図や3次元モデルの使用は、性能は良いが適用可能な環境を限定する。また、高精度なGPSと慣性航法装置を利用した車両のピッチ角推定法も提案されている(Ryu et al., 2002)(Bae et al., 2001)。しかしGPSの使用も、同様に適用可能な環境を限定する。一方、李ら(李他, 2015)は、車載単眼カメラを用いた走行道路に対するピッチ角推定法を提案している。車両が道路平面上を走行するという点に注目し、前後フレーム間の並進ベクトルが走行道路平面に平行である拘束を用いて累積誤差のない推定を示している。しかしここで推定される姿勢角は道路に対する姿勢角でありワールド座標系での絶対値ではない。

本研究では、従来用いられる慣性センサとSLAMの統合に加え、鉛直に建てられた建造物の壁に対する相対姿勢をワールド座標系における絶対姿勢として観測することで、誤差の蓄積を適時補正する姿勢推定法を提案する。主に、時々刻々での推定精度を重視しており、ループクローズによる過去の軌跡の補正是本研究の趣旨と一致しない。なお、本手法は、一般的な建造物の壁がおおよそ鉛直に建てられていることを利用し、事前環境地図を必要としない。屋外での実機走行実験によって本手法の有用性を示す。

## 2. 建造物の壁に対する相対姿勢を用いた姿勢推定

本章では建造物の壁に対する相対姿勢を用いた姿勢推定について述べる。ロボットが静止している状態でIMUによって初期姿勢を求め、それに対して角速度センサを用いたデッドレコニングおよびSLAMで推定される相対姿勢変化を積算する。鉛直面を観測した場合は、それらを用いてロボット座標系での重力ベクトルを推定し蓄積誤差を補正する。なお、この推定姿勢はカルマンフィルタの観測値として統合される。

### 2.1 変数定義

#### 2.1.1 座標系の定義

座標系を以下に定義する。

- ロボット座標系

ロボットに固定され、進行方向をX軸の正とする右手直交座標系とする。各軸をX<sub>r</sub>, Y<sub>r</sub>, Z<sub>r</sub>軸とする。

- ワールド座標系

初期姿勢位置において原点がロボット座標系と一致し、重力方向をZ軸の負とする右手直交座標系とする。各軸をX<sub>w</sub>, Y<sub>w</sub>, Z<sub>w</sub>軸とする。

#### 2.1.2 拡張カルマンフィルタで用いる変数の定義

第2.2節および第2.4節で拡張カルマンフィルタを用いるため、表1に使用する変数の定義を示す。本論文内の拡張カルマンフィルタの記述ではこれらの変数を共通して用いる。

#### 2.1.3 姿勢角の定義

ワールド座標系のX<sub>w</sub>軸, Y<sub>w</sub>軸, Z<sub>w</sub>軸まわりの姿勢角をそれぞれ $\phi$ ,  $\theta$ ,  $\psi$ とする。また、姿勢角の定義域は $[-\pi, \pi]$ であり、数値的に不連続であることを考慮する必要があるが、具体的な処理の記述は省略する。

Table 1 Nomenclature.

$k$	discrete time step
$f, h$	non-linear process and measurement model function
$\mathbf{x}$	state vector
$\mathbf{z}$	measurement vector

## 2.2 IMU を用いた初期姿勢推定

ロボットが静止している状態での IMU の出力値を用いてワールド座標系におけるロボットの初期姿勢を推定する。また角速度センサのバイアス  $\omega_{bias}$  も推定する。初期姿勢の推定には拡張カルマンフィルタを用い、カルマンフィルタの分散が収束するまでの時間、IMU を静止させておく必要がある。式 (1) ~ (4) でそれぞれ示される状態ベクトル、観測ベクトル、状態方程式、観測方程式を用いて、IMU の出力値を取得するたびに予測、観測を行う。ロボットは静止状態であるため、ここでは  $\psi = 0$  で一定とする。座標変換行列を  $\mathbf{R}$ 、ワールド座標系での重力ベクトルを  $\mathbf{G}_{world}$  とする。加速度の平均値を  $\alpha_{ave}$ 、角速度の平均値を  $\omega_{ave}$  とする。例えば  $\alpha_{ave,X_r,k}$  は  $k$  ステップでの  $X_r$  軸方向の平均加速度値を表す。

$$\mathbf{x}_k = \begin{pmatrix} \phi_k & \theta_k \end{pmatrix}^T \quad (1)$$

$$\mathbf{z}_k = \begin{pmatrix} \alpha_{ave,X_r,k} & \alpha_{ave,Y_r,k} & \alpha_{ave,Z_r,k} \end{pmatrix}^T \quad (2)$$

$$f(\mathbf{x}_k) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \phi_{k-1} \\ \theta_{k-1} \end{pmatrix} \quad (3)$$

$$h(\mathbf{x}_k) = \mathbf{R}_k \mathbf{G}_{world} = \mathbf{R}_k \begin{pmatrix} 0 \\ 0 \\ G_{world,Z_w} \end{pmatrix} = \begin{pmatrix} -G_{world,Z_w} \sin \theta \\ G_{world,Z_w} \sin \phi \cos \theta \\ G_{world,Z_w} \cos \phi \cos \theta \end{pmatrix} \quad (4)$$

$$\omega_{bias} = \omega_{ave} \quad (5)$$

## 2.3 壁面を用いた推定姿勢補正

### 2.3.1 主法線抽出

測距センサで得られる点群を用いて、各注目点の近傍局所点群に主成分分析 (Pauly et al., 2002) を適用することで法線ベクトルを算出する。近傍点群は、kd-tree(Bentley, 1975) を用いて指定半径以内の点を探索することで得られる。センサから得られる点群は、センサから遠くなるほど点の密度が粗になるため、注目点とセンサとの距離が大きいほど、探索半径を大きくする。

この法線群から、信頼性の高い鉛直面を持つ法線を以下の条件で抽出する。

- 注目点が持つ近傍点の数が十分多い（密度が高い）。
- 式 (6) で算出される角度  $\beta$  が十分小さい。これは 1 ステップ前の推定重力ベクトル  $\mathbf{G}'_{robot}$  をもとに、観測された法線  $\mathbf{N}$  が鉛直面を持つかを判定するものである。

$$\beta = \left| \cos^{-1} \frac{\mathbf{N} \cdot \mathbf{G}'_{robot}}{\|\mathbf{N}\| \|\mathbf{G}'_{robot}\|} - \frac{\pi}{2} \right| \quad (6)$$

- 法線に垂直な平面と近傍点群の二乗誤差が十分小さい（平面度が高い）。

抽出された法線群でガウス球 (Horn, 1984) を生成し、球内の点群に対してユークリッド距離に基づくクラスタ分析を適用する。ガウス球とは、各点が保有する法線ベクトルの各成分をユークリッド空間内に再配置することで得られる点群であり、この操作は一般にガウス写像と呼ばれる。なお本手法では、計算処理に含まれる点群を増やして精度を高めるため、適当な法線を反転させガウス球をさらに半球のみに反映している。つまり、平行で向

かい合う平面は同じ法線ベクトルを持つようとする。クラスタリング終了後に、メンバが閾値より少ないクラスタは外れ値として除去する。各クラスタが持つ点群の重心を算出し、それらの位置ベクトルを主法線（支配的な法線）として用いる。

### 2.3.2 重力ベクトル推定

クラスタリングされた主法線（ガウス球内の点）を用いてロボット座標系における単位重力ベクトル  $\mathbf{G}_{\text{robot}}$  を推定する。主法線の数に応じて以下のように算出する。ただし、以下のように外積などで算出されるベクトルが、本来の重力ベクトルと方向が逆になる場合もある。そのため、算出後に、1ステップ前の重力方向をもとに、算出したベクトルの向きを判定する必要がある。

- 主法線の数：3 以上

主法線をガウス球における点として扱い、この点群に対する主成分分析で得た法線を重力ベクトルとする。ただし、クラスタが持つ点の数で重み付けを行う。

- 主法線の数：2

2 本の主法線ベクトル  $\mathbf{N}_1, \mathbf{N}_2$  の外積を重力方向ベクトルとする。図 1a に推定の様子を示す。

$$\mathbf{G}_{\text{robot}} = \mathbf{N}_1 \times \mathbf{N}_2 \quad (7)$$

- 主法線の数：1

ワールド座標系における絶対的な姿勢を得るには、平行でない鉛直面が 2 面以上必要であるが、1ステップ前の推定重力ベクトル  $\mathbf{G}'_{\text{robot}}$  を用いて、式 (8) のように部分的に推定重力ベクトルを補正し算出する。観測された法線ベクトルを  $\mathbf{N}$  とする。図 1b に推定の様子を示す。

$$\mathbf{G}_{\text{robot}} = \frac{\mathbf{G}'_{\text{robot}} - (\mathbf{G}'_{\text{robot}} \cdot \mathbf{N})\mathbf{N}}{\|\mathbf{G}'_{\text{robot}} - (\mathbf{G}'_{\text{robot}} \cdot \mathbf{N})\mathbf{N}\|} \quad (8)$$

- 主法線の数：0

重力方向を推定することが出来ない。

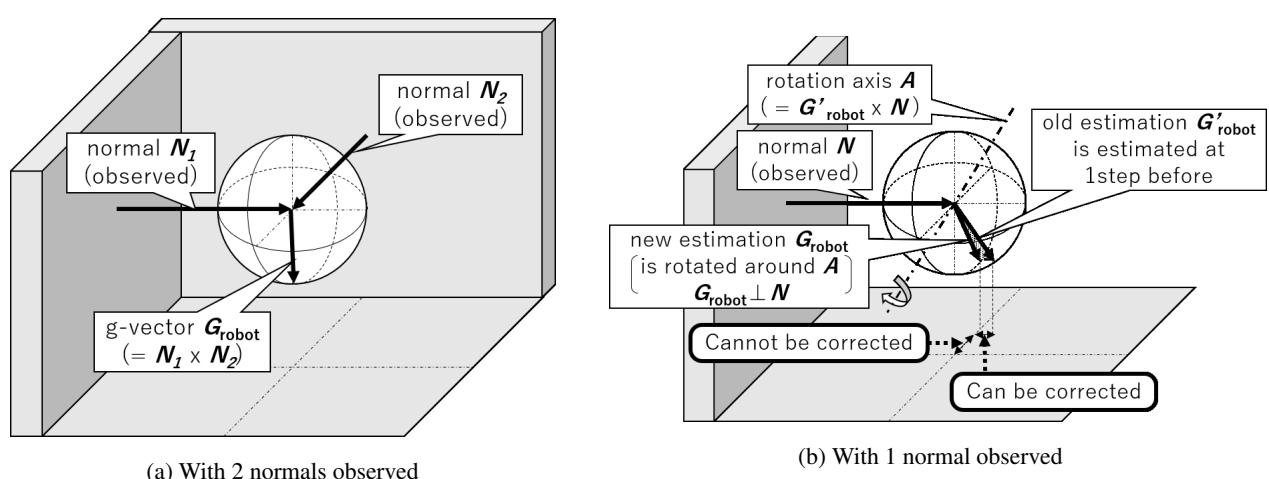


Fig. 1 In (a), the gravity vector  $\mathbf{G}_{\text{robot}}$  is estimated by the cross product of the two observed normals  $\mathbf{N}_1, \mathbf{N}_2$ .  $\mathbf{N}_1, \mathbf{N}_2$  are normals which are extracted and clustered, and they are not parallel. In (b), estimating the gravity vector  $\mathbf{G}_{\text{robot}}$  is supposed to need two or more vectors which are not parallel to each other. The proposed method partially corrects the estimation of the gravity vector by using  $\mathbf{G}'_{\text{robot}}$  which is estimated one step before.

### 2.3.3 推定姿勢補正

重力ベクトルの推定誤差が、推定姿勢の誤差であることを用いて補正する。なお計算は四元数（quaternion）を用いて行う。ただし、第2.2節や第2.4節で記述されるカルマンフィルタでは、姿勢をロール、ピッチ、ヨーで扱っているため、四元数に変換する必要がある。補正後の姿勢を表す四元数  $\mathbf{q}_{\text{pose.wall}}$  の算出を以下に示す。1ステップ前の推定重力ベクトル  $\mathbf{G}'_{\text{robot}}$  と、第2.3.2項で推定した推定重力ベクトル  $\mathbf{G}_{\text{robot}}$ において、この2ベクトル間の回転を表す四元数  $\mathbf{q}_{\text{error}}$  は、

$$\begin{aligned}\mathbf{q}_{\text{error}} &= \begin{pmatrix} q_{\text{error},x} & q_{\text{error},y} & q_{\text{error},z} & q_{\text{error},w} \end{pmatrix} \\ &= \begin{pmatrix} C_{X_r} \sin \gamma/2 & C_{Y_r} \sin \gamma/2 & C_{Z_r} \sin \gamma/2 & \cos \gamma/2 \end{pmatrix}\end{aligned}\quad (9)$$

$$\gamma = \cos^{-1} \frac{\mathbf{G}'_{\text{robot}} \cdot \mathbf{G}_{\text{robot}}}{\|\mathbf{G}'_{\text{robot}}\| \|\mathbf{G}_{\text{robot}}\|} \quad (10)$$

$$\mathbf{C} = \begin{pmatrix} C_{X_r} & C_{Y_r} & C_{Z_r} \end{pmatrix}^T = \frac{\mathbf{G}'_{\text{robot}} \times \mathbf{G}_{\text{robot}}}{\|\mathbf{G}'_{\text{robot}}\| \|\mathbf{G}_{\text{robot}}\|} \quad (11)$$

推定重力ベクトル  $\mathbf{G}'_{\text{robot}}$  に対応する推定姿勢を表す四元数を  $\mathbf{q}'_{\text{pose}}$  としたとき、次のように補正される。

$$\mathbf{q}_{\text{pose.wall}} = \mathbf{q}'_{\text{pose}} \mathbf{q}_{\text{error}} \quad (12)$$

$\mathbf{q}_{\text{pose.wall}}$  および  $\mathbf{q}'_{\text{pose}}$  はワールド座標系での姿勢であり、 $\mathbf{q}_{\text{error}}$  はロボット座標系での誤差（回転）である。

### 2.3.4 キャリブレーション

距離センサの取り付け誤差があった場合、それに合わせて点群を回転させる必要がある。ロボットが初期位置姿勢に静止している状態でキャリブレーションを行う。第2.2節で推定した初期姿勢を表す四元数  $\mathbf{q}_{\text{pose.ini}}$  に対する、第2.3.3項で推定した姿勢を表す四元数  $\mathbf{q}_{\text{pose.wall}}$  の誤差を取り付け誤差  $\mathbf{q}_{\text{misalignment}}$  とする。

$$\mathbf{q}_{\text{misalignment}} = \mathbf{q}_{\text{pose.wall}} \mathbf{q}_{\text{pose.ini}}^{-1} \quad (13)$$

初期姿勢推定後は、点群に対して回転  $\mathbf{q}_{\text{misalignment}}$  を適用させて上記の重力ベクトル推定を行う。

## 2.4 拡張カルマンフィルタによる統合

ここでは車輪型ロボットで人工環境を走行することを想定しており、各姿勢角度変化が急激なものでなく、カルマンフィルタで十分扱えると仮定する。IMUを用いたデッドレコニングで推定する姿勢を予測、SLAMで推定する姿勢を観測1、壁面に対する相対姿勢を観測2とし、拡張カルマンフィルタで統合する。

$$\mathbf{x}_k = \begin{pmatrix} \phi_k & \theta_k & \psi_k \end{pmatrix}^T \quad (14)$$

### • 予測（デッドレコニング）

ロボット座標系における角速度を、ワールド座標系へ座標変換しレート積分を行う。ただし、センサのバイアスを考慮している。

$$f(\mathbf{x}_{k-1}) = \begin{pmatrix} \phi_{k-1} \\ \theta_{k-1} \\ \psi_{k-1} \end{pmatrix} + \begin{pmatrix} 1 & \sin \phi_{k-1} \tan \theta_{k-1} & \cos \phi_{k-1} \tan \theta_{k-1} \\ 0 & \cos \phi_{k-1} & -\sin \phi_{k-1} \\ 0 & \sin \phi_{k-1} \sec \theta_{k-1} & \cos \phi_{k-1} \sec \theta_{k-1} \end{pmatrix} \begin{pmatrix} \omega_{X_r,k-1} - \omega_{\text{bias},X_r} \\ \omega_{Y_r,k-1} - \omega_{\text{bias},Y_r} \\ \omega_{Z_r,k-1} - \omega_{\text{bias},Z_r} \end{pmatrix} dt \quad (15)$$

- 観測 1 (SLAM)

SLAMによって得られる姿勢をそのまま観測するのではなく、1ステップ前に SLAMで推定された姿勢との相対変化を用いることで、観測2を推定結果に反映しやすくする。

$$\mathbf{z}_k = \begin{pmatrix} \phi_{k-1} + (\phi_{\text{slam},k} - \phi_{\text{slam},k-1}) \\ \theta_{k-1} + (\theta_{\text{slam},k} - \theta_{\text{slam},k-1}) \\ \psi_{k-1} + (\psi_{\text{slam},k} - \psi_{\text{slam},k-1}) \end{pmatrix} \quad (16)$$

$$h(\mathbf{x}_k) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \phi_k \\ \theta_k \\ \psi_k \end{pmatrix} \quad (17)$$

- 観測 2 (壁面観測)

上記2つのステップでは初期姿勢に対する相対姿勢を推定しているのに対し、ここではワールド座標系での絶対姿勢を直接観測することができる。式(12)で算出される四元数  $\mathbf{q}_{\text{pose.wall}}$  をワールド座標系のロール、ピッチに変換し観測する。第2.3.3項で場合分けされるように、主法線の数に応じて共分散を変化させる。

$$\mathbf{z}_k = \begin{pmatrix} \tan^{-1} \frac{2(q_{\text{pose.wall},w}q_{\text{pose.wall},x} + q_{\text{pose.wall},y}q_{\text{pose.wall},z})}{q_{\text{pose.wall},w}^2 - q_{\text{pose.wall},x}^2 - q_{\text{pose.wall},y}^2 + q_{\text{pose.wall},z}^2} \\ \sin^{-1} 2(q_{\text{pose.wall},w}q_{\text{pose.wall},y} - q_{\text{pose.wall},x}q_{\text{pose.wall},z}) \end{pmatrix} \quad (18)$$

$$h(\mathbf{x}_k) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \phi_k \\ \theta_k \\ \psi_k \end{pmatrix} \quad (19)$$

### 3. 評価実験

#### 3.1 実験 (A)

##### 3.1.1 実験 (A) 概要

起伏のあるコースで移動ロボットを走行させ、提案手法および比較手法で、時々刻々のロボットの自己姿勢を推定した。走行中の推定姿勢を評価することは難しいため、推定姿勢と並進速度を統合することで、走行中の推定姿勢誤差を、走行終了時の推定位置誤差に置き換えて評価した。つまり、同じ並進速度を用いた場合、姿勢の推定誤差が小さいほど、推定位置（軌跡）の誤差が小さくなると仮定した。

##### 3.1.2 実験 (A) 環境

本実験での提案手法のシステム図を図2に示す。図中、Staticは第2.2節の初期姿勢推定のシステム、Dynamicは走行開始後のシステムをそれぞれ表している。カルマンフィルタの観測で用いるSLAMとして、LSD-SLAM(Engel et al., 2014)を用いた。LSD-SLAMとは単眼カメラSLAMである。推定のレートの速さとスケールに依存しないことが姿勢推定に適していると判断し選定した。図中、Integrationでは式(4)で用いた座標変換行列  $\mathbf{R}$  を用い、式(20)のように座標変換を行い、姿勢と並進速度が統合される。 $\mathbf{V}_{\text{robot}}$  はロボット座標系での並進速度ベクトル、 $\mathbf{L}_{\text{world}}$  はワールド座標系でのロボットの推定位置ベクトルである。

$$\begin{aligned} \mathbf{L}_{\text{world},k} &= \mathbf{L}_{\text{world},k-1} + \mathbf{R}_k^{-1} \mathbf{V}_{\text{robot},k} dt \\ &= \mathbf{L}_{\text{world},k-1} + \mathbf{R}_k^T \mathbf{V}_{\text{robot},k} dt \\ &= \begin{pmatrix} L_{X_w,k-1} \\ L_{Y_w,k-1} \\ L_{Z_w,k-1} \end{pmatrix} + \mathbf{R}_k^T \begin{pmatrix} V_{\text{robot},X_r,k} \\ 0 \\ 0 \end{pmatrix} dt \end{aligned} \quad (20)$$

本実験では、統合される並進速度としてホイールオドメトリで得られる並進速度を用いた。

姿勢推定の比較手法として、ジャイロスコープによるデッドレコニングと、単独の LSD-SLAM を用いた。提案手法と同様に、これらの手法で姿勢をそれぞれ推定し、ホイールエンコーダの並進速度と統合した。なお、本研究の趣旨より、LSD-SLAM でのループクローズは行っていない。

図 3 に実験で使用する移動ロボットを示す。ロボットに搭載し、使用したセンサは、Velodyne HDL-32E、Xsens MTi30、RealSense D435、ホイールエンコーダである。RealSense D435 は RGB-D カメラであるが、LSD-SLAM のために画像のみを利用したため、深度情報は利用していない。評価に用いる真値として、走行開始時と走行終了時における、モーションキャプチャで得られる位置姿勢を用いた。本実験では Vicon Vero v1.3X を用いた。図 4 に、明治大学生田キャンパス内の実験環境の航空写真を示す。走行コースは、一周約 250m、高低差約 3m であり、図 4 に示すように 3 周走行した。

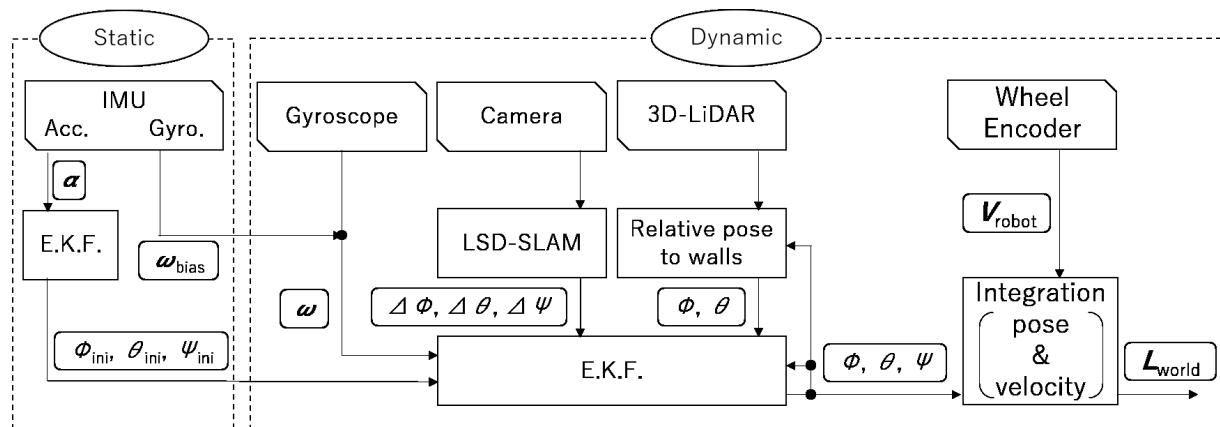


Fig. 2 A system configuration diagram of the proposed method for the experiment is shown here. “Static” part represents the system which runs when the robot has no move. “Dynamic” part represents the system after the robot starts driving. Estimating pose with vertical walls is referred to as “Relative pose to walls” in this figure, and the pose is merged through E.K.F. with other estimations. In “Integration” part, estimated pose(orientation) and linear velocity are integrated.

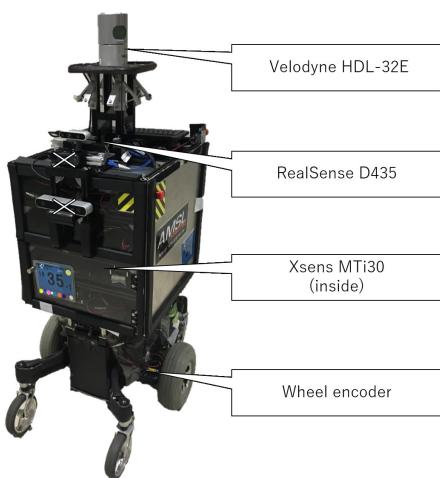


Fig. 3 The experimental mobile robot has 3D-LIDAR, camera, IMU and wheel encoders. RealSense D435 is RGBD camera, however this is just for LSD-SLAM and depth information is not used in the experiment.

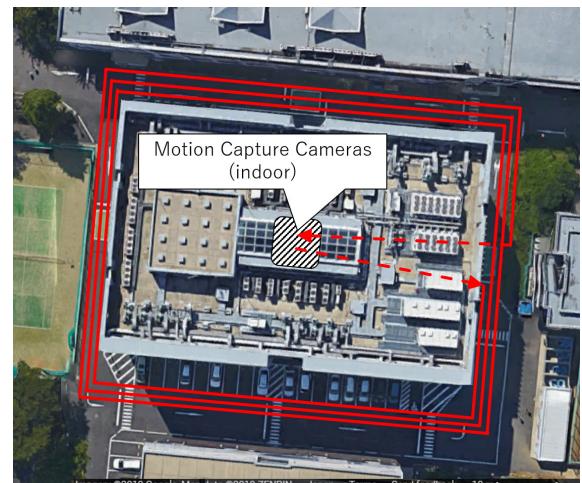


Fig. 4 An aerial shot of the experimental environment is shown here. Motion capture cameras locate on indoor area of a building. The robot starts driving at that area and comes back there. The course the robot drives has some slopes.

### 3.1.3 実験 (A) 結果

表2に、各手法での推定位置姿勢の誤差を示す。表中の誤差は、ワールド座標系の座標軸を基準とした位置および姿勢角の誤差である。また、Euc.dist.は、ユークリッド距離で表した位置推定誤差である。ユークリッド距離で表した位置推定誤差に関して、比較手法に比べて、提案手法の値が小さいことから、走行中の姿勢推定の誤差も提案手法の方が小さいと言える。図5に、各手法で推定されたロボットの軌跡を示す。この図より、同じコースを3周していることをふまえると、提案手法の推定軌跡の誤差がより小さいことが視覚的にも分かる。特に、 $Z_w$ 軸(鉛直)方向の並進誤差が小さく抑えられており、壁面を用いた補正による結果だと考察する。第2.3.2項で記述したように、環境によって主法線の数が異なり、処理も異なる。表3に、この実験コースでの走行における、主法線の数に応じた壁情報での補正回数の割合を示す。表4に、同様の実験環境でさらに実験を繰り返した際の、提案手法による推定誤差の平均値および分散を示す。ただし、追加で行った実験では1周のみの走行であり、それを5回行った。

Table 2 Errors of position and pose estimations.

error in...	X <sub>w</sub> [m]	Y <sub>w</sub> [m]	Z <sub>w</sub> [m]	Euc.dist.[m]	ϕ[deg]	θ[deg]	ψ[deg]
proposed method	+1.8843	-0.6136	+1.9233	2.762	-2.069	-0.095	-0.300
LSD-SLAM	+4.8471	-2.3376	+9.0677	10.544	+6.922	-8.721	+18.257
Gyrodometry	+2.0761	-0.2382	-2.6726	3.393	-2.984	+6.970	-0.963

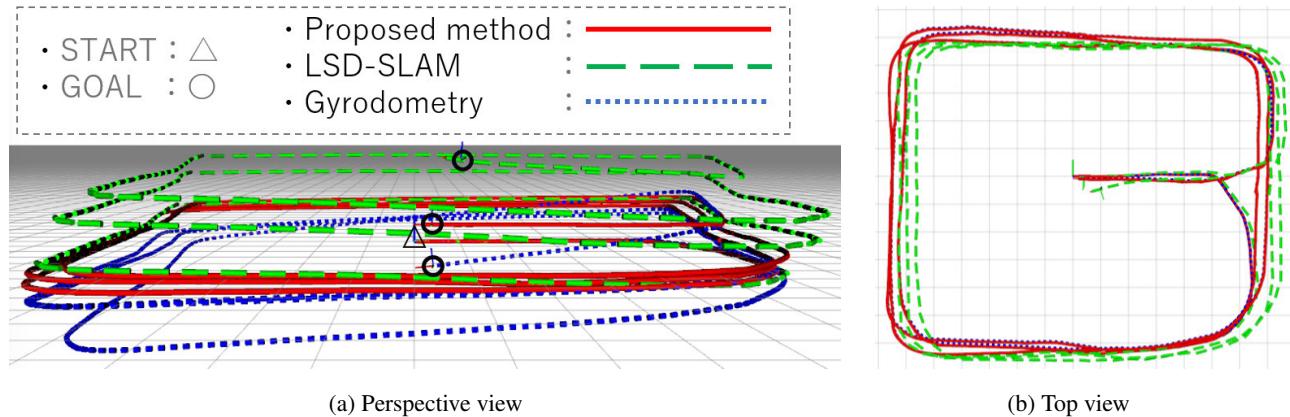


Fig. 5 Estimated trajectories of the robot are shown here. It is visually found that trajectories estimated by Gyrodometry and LSD-SLAM respectively have larger translational errors in  $Z_w$  axis.

Table 3 Ratio of times of each correcting type to number of scans.

n: number of dominative normals	$n \geq 3$	$n = 2$	$n = 1$	$n = 0$
ratio[%]	0.09	39.06	60.84	0

Table 4 Average and variance of the proposed method in 1-round experiments.

	X <sub>w</sub> [m]	Y <sub>w</sub> [m]	Z <sub>w</sub> [m]	Euc.dist.[m]	ϕ[deg]	θ[deg]	ψ[deg]
ave.	0.5119	0.4260	-0.0748	0.8153	0.018	0.006	3.032
var.	0.1817	0.1967	0.0305	0.1030	0.059	0.120	5.034

### 3.2 実験 (B)

#### 3.2.1 実験 (B) 概要

鉛直でない平面が提案手法に与える影響を、簡易的な実験によって検証した。鉛直面で囲まれた屋内空間に、鉛直でない平面を持つ障害物を設置し、第2.3節で記述したように鉛直面を観測し姿勢を推定した。ただし、ロボットは静止させ、IMUを使ったデッドレコニングおよびSLAMは行っていない。また、第2.3.4項のキャリブレーションは行っていない。

#### 3.2.2 実験 (B) 環境

実験 (A)と同じロボットを用いて実験を行った。障害物として、表5のような掲示用のボードとアルミ板を使用し、傾斜角や障害物の種類および数を変えて比較した。図6に実験の様子、図7に実験環境で取得された点群をそれぞれ示す。実験環境が、屋内であり十分整備されているため、ロボットの姿勢角の真値は $\phi_{\text{true}} = 0$ ,  $\theta_{\text{true}} = 0$ として評価に用いた。

#### 3.2.3 実験 (B) 結果

表6に実験 (B) の結果を示す。表内の誤差は、1分間の平均である。また、実験結果の様子の例として、実験no.3の3次元点群および法線群を図8に示す。なお図中のextracted normalは、第2.3.1項で設定される条件を満たし、信頼性が高いと判定された法線、clustered normalは主法線を表す。障害物の傾きが大きくなるほど、推定姿勢角の誤差は大きくなる。ただし、式(6)の $\beta$ が閾値を超えた場合、実験no.4のように障害物の影響がなくなる。実験no.4の様子を図9に示す。また、障害物が持つ平面の面積が大きいほど、推定姿勢角の誤差は大きくなる。これは第2.3.2項の通り、観測された法線の数が3以上の場合、クラスタが持つメンバの数で重み付けを行っていることに由来していると考える。ただし、メンバの数が閾値を下回った場合外れ値とするため、実験no.5のように、障害物の影響がなくなる。実験no.5の様子を図10に示す。信頼性の高い法線であるかの判定や、クラスリングの外れ値であるかの判定の基準はパラメータで調節している。判定基準を厳しくすれば、鉛直面の誤認識を防げるが、本来の目的である補正がかかりづらくなってしまう。想定される環境に応じて、適当に設定する必要がある。

Table 5 Detail of obstacles.

name	dimention	description
small plane	400 mm × 600 mm	aluminum board
large plane	1530 mm × 900 mm	bulletelin board

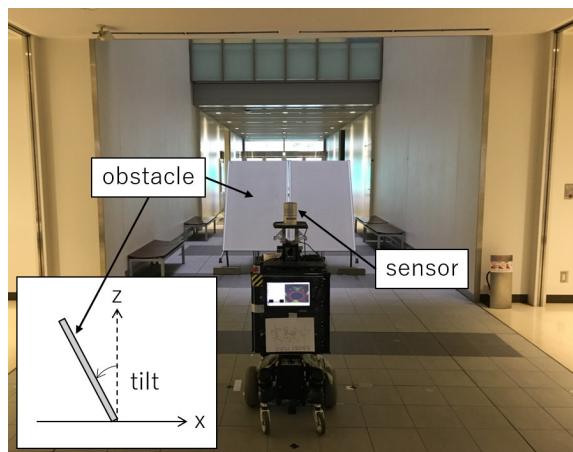


Fig. 6 The Environment of the experiment(B).

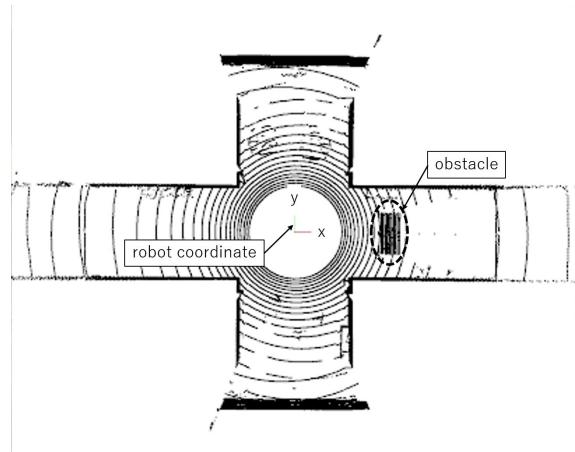


Fig. 7 Point cloud of the experimental environment(B).

Table 6 Errors of pose estimations with obstacles which are not vertical planes.

no.	obstacle type	tilt[deg]	roll error[deg]	pitch error[deg]	recital
1	no obstacle	-	+0.103	-0.831	-
2	large plane×1	15	+1.24	+1.57	-
3	large plane×1	30	+1.89	+3.01	-
4	large plane×1	45	+0.104	-0.831	out of the angle range
5	small plane×1	30	+0.113	-0.832	outlier
6	large plane×2	30	+3.96	+7.38	-

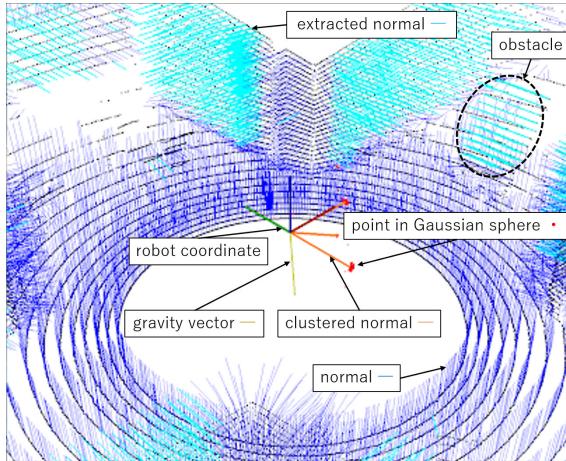


Fig. 8 Point and normal cloud in the experiment no.3 are shown here. Extracted normals are picked out from the original normals, with conditions for judging whether a normal has a vertical plane. These extracted normals are projected to a Gaussian sphere. And the points in the Gaussian sphere are clustered.

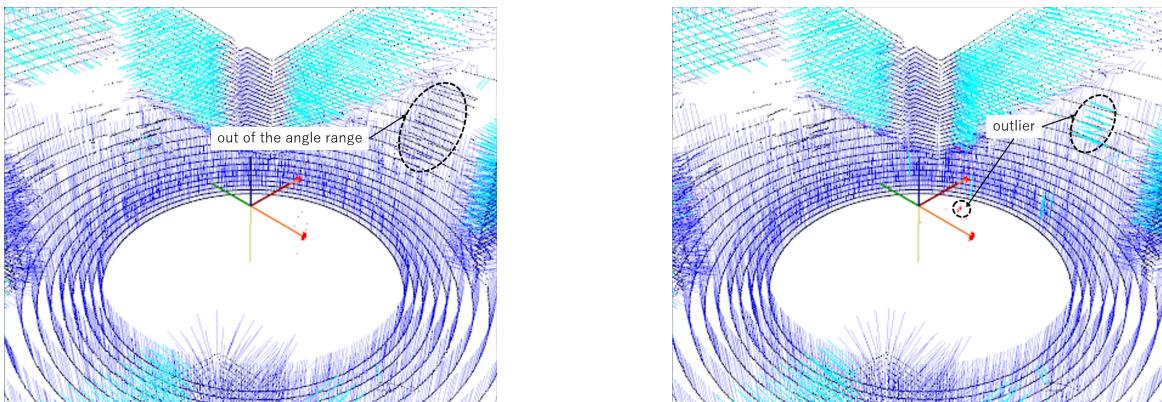


Fig. 9 Point and normal cloud of the experiment no.4 are shown here. There is no influence for the pose estimation when experimental obstacles are tilted more than the threshold which the proposed method has.

Fig. 10 Point and normal cloud of the experiment no.5 are shown here. There is no influence for the pose estimation when experimental obstacles are small enough to be recognized as outliers in the clustering step.

#### 4. 結 言

従来の慣性センサと SLAM による移動ロボットの姿勢推定に加え、鉛直に建てられている建造物の壁に対する相対姿勢を利用する姿勢推定法を提案した。実験 (A) の結果より、鉛直面を用いることで走行中における姿勢推定の蓄積誤差を小さく抑えるという本提案手法の有用性が示された。また、実験 (B) の結果より、鉛直でない平面に対する処理の有効性も示された。ただし、想定される環境に応じて判定基準を適切に設定する必要がある。

謝 辞

本研究の一部は、国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の、次世代人工知能・ロボット中核技術開発事業による支援を受けた。ここに篤く御礼申し上げる。

文 献

- Bae, H. S., Ryu, J. and Gerdes, J. C., Road grade and vehicle parameter estimation for longitudinal control using GPS, Proceedings of IEEE Conference on Intelligent Transportation Systems (2001), pp.166–171.
- 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 the IEEE International Conference on Intelligent Robots and Systems, Vol.3 (2003), pp.2743–2748.
- Engel, J., Koltun, V. and Cremers, D., Direct sparse odometry, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.40, No.3 (2018), pp.611–625.
- Engel, J., Stueckler, J. and Cremers, D., LSD-SLAM: large-scale direct monocular SLAM, European Conference on Computer Vision (2014), pp.834–849.
- 後藤翼, Pathak, S., 池勇勲, 藤井浩光, 山下淳, 浅間一, 人工物環境における全天球カメラの位置姿勢推定のための直線特徴に基づく3D-2Dマッチング, 精密工学会誌, Vol.83, No.12 (2017), pp.1209–1215.
- Hata, A. Y., Osorio, F. S. and Wolf, D. F., Robust curb detection and vehicle localization in urban environments, IEEE Intelligent Vehicles Symposium Proceedings (2014), pp.1257–1262.
- Horn, B. K. P., Extended gaussian images, Proceedings of the IEEE, Vol.72, No.12 (1984), pp.1671–1686.
- Kerl, C., Sturm, J. and Cremers, D., Dense visual SLAM for RGB-D cameras, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2013), pp.2100–2106.
- Kwon, S., Kim, S. and Yu, J., Tilting-type balancing mobile robot platform for enhancing lateral stability, IEEE/ASME transactions on mechatronics, Vol.20, No.3 (2015), pp.1470–1481.
- 李博, 張曉林, 佐藤誠, 車間距離計測のための車載単眼カメラを用いたピッチ角推定, 映像情報メディア学会誌 Vol.69, No.4 (2015), pp.169–176.
- Magnusson, M., Lilienthal, A. and Duckett, T., Scan registration for autonomous mining vehicles using 3D-NDT, Journal of Field Robotics, Vol.24, No.10 (2007), pp.803–827.
- Pauly, M., Gross, M. and Kobbelt, L.P., Efficient simplification of point-sampled surfaces, Proceedings of the conference on Visualization (2002), pp.163–170.
- 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.
- Ramalingam, S., Bouaziz, S., Sturm, P. and Brand, M., SKYLINE2GPS: localization in urban canyons using omni-skylines, IEEE/RSJ International Conference on Intelligent Robots and Systems (2010), pp.3816–3823.
- Rusinkiewicz, S. and Levoy, M., Efficient variants of the ICP algorithm, Proceedings Third International Conference on 3-D Digital Imaging and Modeling (2001), pp.145–152.
- Ryu, J., Rossetter, E. J. and Gerdes, J. C., Vehicle sideslip and roll parameter estimation using GPS, Proceedings of the Symposium on Advanced Vehicle Control (2002), pp.373–380.
- 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講演予稿集 (2014), pp.453–458.
- Stumberg, L.von, Usenko, V. and Cremers, D., Direct sparse visual-inertial odometry using dynamic marginalization, in International Conference on Robotics and Automation (2018), pp.2510–2517.
- Thrun, S., Burgard, W. and Fox, D., Probabilistic robotics (2005), pp.309–336, The MIT Press.
- Zhang, J. and Singh, S., LOAM: lidar odometry and mapping in real-time, in Robotics: Science and Systems Conference (2014), pp.161–195.

## References

- Bae, H. S., Ryu, J. and Gerdes, J. C., Road grade and vehicle parameter estimation for longitudinal control using GPS, Proceedings of IEEE Conference on Intelligent Transportation Systems (2001), pp.166–171.
- 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 the IEEE International Conference on Intelligent Robots and Systems, Vol.3 (2003), pp.2743–2748.
- Engel, J., Koltun, V. and Cremers, D., Direct sparse odometry, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.40, No.3 (2018), pp.611–625.
- Engel, J., Stueckler, J. and Cremers, D., LSD-SLAM: large-scale direct monocular SLAM, European Conference on Computer Vision (2014), pp.834–849.
- Goto, T., Pathak, S., Ji, Y., Fujii, H., Yamashita, A. and Asawa, H., 3D-2D matching of line features for spherical camera localization in man-made environment, Journal of the Japan Society for Precision Engineering, Vol.83, No.12 (2017), pp.1209–1215 (in Japanese).
- Hata, A. Y., Osorio, F. S. and Wolf, D. F., Robust curb detection and vehicle localization in urban environments, IEEE Intelligent Vehicles Symposium Proceedings (2014), pp.1257–1262.
- Horn, B. K. P., Extended gaussian images, Proceedings of the IEEE, Vol.72, No.12 (1984), pp.1671–1686.
- Kerl, C., Sturm, J. and Cremers, D., Dense visual SLAM for RGB-D cameras, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2013), pp.2100–2106.
- Kwon, S., Kim, S. and Yu, J., Tilting-type balancing mobile robot platform for enhancing lateral stability, IEEE/ASME transactions on mechatronics, Vol.20, No.3 (2015), pp.1470–1481.
- Li, B., Zhang, X. and Sato, M., Pitch angle estimation using a vehicle mounted monocular camera for vehicle target range measurement, The journal of the Institute of Image Information and Television Engineers, Vol.69, No.4 (2015), pp.169–176 (in Japanese).
- Magnusson, M., Lilienthal, A. and Duckett, T., Scan registration for autonomous mining vehicles using 3D-NDT, Journal of Field Robotics, Vol.24, No.10 (2007), pp.803–827.
- Pauly, M., Gross, M. and Kobbelt, L.P., Efficient simplification of point-sampled surfaces, Proceedings of the conference on Visualization (2002), pp.163–170.
- 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.
- Ramalingam, S., Bouaziz, S., Sturm, P. and Brand, M., SKYLINE2GPS: localization in urban canyons using omni-skylines, IEEE/RSJ International Conference on Intelligent Robots and Systems (2010), pp.3816–3823.
- Rusinkiewicz, S. and Levoy, M., Efficient variants of the ICP algorithm, Proceedings Third International Conference on 3-D Digital Imaging and Modeling (2001), pp.145–152.
- Ryu, J., Rossetter, E. J. and Gerdes, J. C., Vehicle sideslip and roll parameter estimation using GPS, Proceedings of the Symposium on Advanced Vehicle Control (2002), pp.373–380.
- 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).
- Stumberg, L.von, Usenko, V. and Cremers, D., Direct sparse visual-inertial odometry using dynamic marginalization, in International Conference on Robotics and Automation (2018), pp.2510–2517.
- Thrun, S., Burgard, W. and Fox, D., Probabilistic robotics (2005), pp.309–336, The MIT Press.
- Zhang, J. and Singh, S., LOAM: lidar odometry and mapping in real-time, in Robotics: Science and Systems Conference (2014), pp.161–195.

RESEARCH

# EKF-based self-attitude estimation with DNN learning landscape information

Ryota Ozaki\* and Yoji Kuroda

## Abstract

This paper presents an EKF-based self-attitude estimation with a DNN (deep neural network) learning landscape information. The method integrates gyroscopic angular velocity and DNN inference in the EKF. The DNN predicts a gravity vector in a 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 corresponded gravity vectors. The dataset is collected in a flight simulator because we can easily obtain various gravity vectors, although the method is not only for UAVs. Using a simulator breaks the limitation of amount of collecting data with ground truth. The validation shows the network can predict the gravity vector from only a single shot image. It also shows that the covariance matrix expresses the uncertainty of the inference. The covariance matrix is used for integrating the inference in the EKF. Flight data of a drone is also recorded in the simulator, and the EKF-based method is tested with it. It shows the method suppresses accumulative error by integrating the network outputs.

**Keywords:** Attitude estimation; Mobile robotics; Deep learning; Extended Kalman filter

## 1 INTRODUCTION

Estimating attitude of a robot or a UAV 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 accelerometers and gyroscopes. However, mobile robots have their own acceleration. Moreover, on-road robots also receive pulses from the ground, and UAVs suffer from vibration of their rotors. These need to be filtered out from the accelerometer. On the other hand, integration of gyroscopic angular rate has problems 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 registers point clouds by ICP[3], NDT[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. Some methods use prior information such as 3D maps for estimating self-pose [7]. These methods correct the error by matching the prior information

against data from the sensor. However, they work only in environments where maps are available. Moreover, creating a map is time-consuming and also requires update. Some methods[8, 9] estimate attitude under Manhattan world assumption. They assume that planes and edges in the environment are orthogonal to each other. It helps achieving drift-free estimation. However, it is difficult for this kind of methods to avoid being affected by objects which do not satisfy the assumption.

Deep learning has been used for attitude estimation in recent years. In [10], IMU-based odometry by end-to-end learning has been proposed. In [11], a deep neural network identifies the measurement 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, context, and landscape information; for example, most of all artificial buildings should be built vertically, the sky should be seen when the camera orients upper, and so on. The method does not depend on time sequence since only a single shot image is used to estimate the 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

\*Correspondence: ce192021@meiji.ac.jp

Graduate School of Science and Technology, Meiji University, 1-1-1, Higashimita, Tama-ku, 214-8571 Kanagawa, JP  
Full list of author information is available at the end of the article

cannot express uncertainty of the prediction; for instance, the network outputs estimation even when the camera is all covered by obstacles, when less features are captured, and so on. These outputs with large error worsen estimation when it is used in filter functions such as Kalman filter[15]. Therefore, they should be detected and be rejected before the integration.

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 variance. Only outputs with small variance are absorbed in the EKF (extended Kalman filter) to suppress accumulative error. Moreover, the output covariance matrices are used to adjust the process uncertainty in the EKF. 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 a hand carried phone.
- L2 normalization is applied to the output gravity vector while ReLU is applied in the related work.
- Our network outputs mean and a covariance matrix while the related work directly outputs the gravity vector.
- Our method filters out DNN inferences with large variance before processing the integration in the EKF.
- Our method uses the covariance matrix output from the DNN as update process noise in the EKF.

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

## 2 DNN ESTIMATING GRAVITY VECTOR

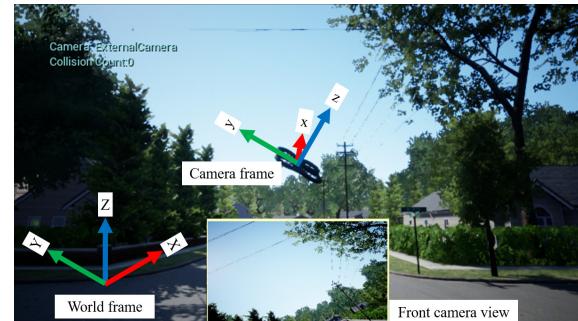
The proposed method makes the DNN learn landscape information for estimating a gravity vector in a camera frame. The gravity is expressed as mean and covariance to consider uncertainty of the prediction.

### 2.1 Coordinate definition

A camera frame is defined as a right-handed coordinate system which is fixed on the camera pose. It is shown in Fig.1.

### 2.2 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 corresponded gravity vectors  $\mathbf{g}$  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  $\phi$  and pitch  $\theta$



**Figure 1: Screenshot of AirSim with coordinate description.** An IMU and a camera are equipped to the drone in the simulator. The purpose of the proposed neural network is estimating a gravity vector in the camera frame from a front camera image.

are limited as [-30 deg, 30 deg], respectively. Fig.2 shows examples of the dataset.

Real data is also collected with the sensors in Fig.3 for test. The stick is hand-carried, and images and linear acceleration vectors measured with the IMU (Xsens MTi-30) are recorded. They are saved only when the stick is shaking less than 0.001 m, 0.1 deg, and when it is at least 5 deg away from the last saved pose. The IMU is regarded as ground truth because it has enough accuracy (within 0.2 deg) in static according to the specification.

### 2.3 Data transformation and augmentation

Input data and label data are transformed, and are augmented in each epoch of training.

#### 2.3.1 Image (input)

The image is randomly rotated for augmenting the roll data. The rotation angle  $\Delta\phi$  is limited as [-10 deg, 10 deg]. The image is resized to  $224 \times 224$ . RGB values are normalized. In this work, this normalization is done following  $mean = (0.5, 0.5, 0.5)$  and  $std = (0.5, 0.5, 0.5)$ . Fig.4 shows an example of the data augmentation.

#### 2.3.2 Gravity vector (label)

A gravity vector in the camera frame is rotated according to  $\Delta\phi$ . 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.

$$\begin{aligned} g_{\text{trans}} &= \text{Rot}_{(\Delta\phi)}^{\text{xyz}} \frac{\mathbf{g}}{|\mathbf{g}|} \\ \text{Rot}_{(\Delta\phi)}^{\text{xyz}} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(-\Delta\phi) & -\sin(-\Delta\phi) \\ 0 & \sin(-\Delta\phi) & \cos(-\Delta\phi) \end{pmatrix} \end{aligned} \quad (1)$$

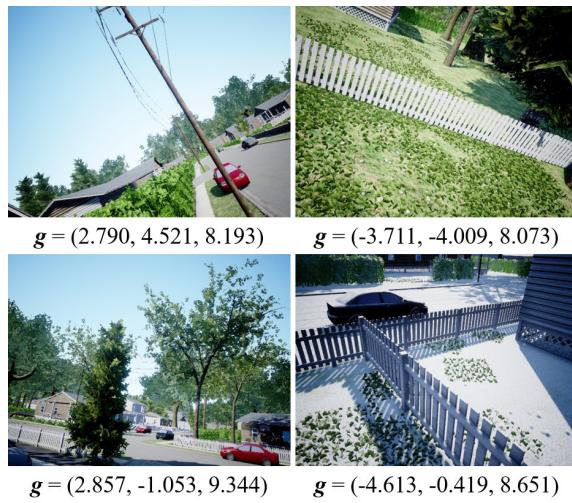


Figure 2: **Examples of datasets.** The dataset consists of images and correspond gravity vectors  $\mathbf{g}$  [ $\text{m/s}^2$ ] in the camera frame. These examples were collected in “Neighborhood” of AirSim. The camera pose and weather parameters are randomized for creating a dataset. An example of the weather parameter is that the road in the lower right image is covered in snow.

## 2.4 Network

The proposed DNN is shown in Fig.5. It consists of CNN layers and FC layers. Its input is a resized image, and its outputs are a mean vector of a gravity vector and a covariance matrix. Technically, the output of the FC layers is  $(\mu_{gx}, \mu_{gy}, \mu_{gz}, L_0, \dots, L_5)$ , and the mean vector  $\mu$  and the covariance matrix  $\Sigma$  are computed as Eq.2, 3, respectively. Since the lower-triangular matrix  $L$  is required to have positive-valued diagonal entries, an exponential function is applied to the diagonal elements.

$$\boldsymbol{\mu} = \frac{(\mu_{gx}, \mu_{gy}, \mu_{gz})^T}{|(\mu_{gx}, \mu_{gy}, \mu_{gz})^T|} \quad (2)$$

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

It is expected that the CNN layers lean extracting features such as edges, and the FC layers learn landscape information. Feature module of VGG16[16] pre-trained on ImageNet[17] is adopted as the CNN layers of the proposed method. All layers, except the final output layer, use the ReLU function[18] as activation function. All FC layers, except the final output layer, use the 10% Dropout[19].

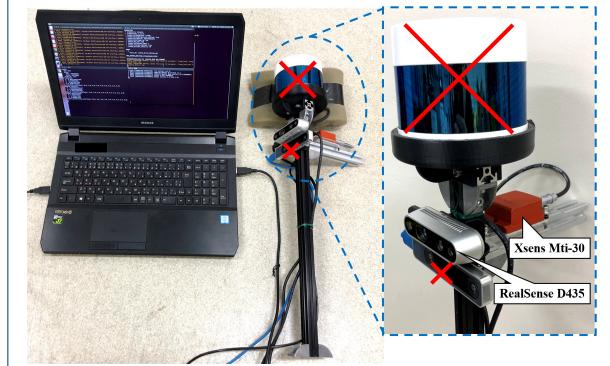


Figure 3: **Sensor stick.** Images and acceleration are recorded with this stick when it is still. The judge whether it is still is processed by programming. Note that depth information is not used in this study although RealSense D435 is a RGB-D camera.



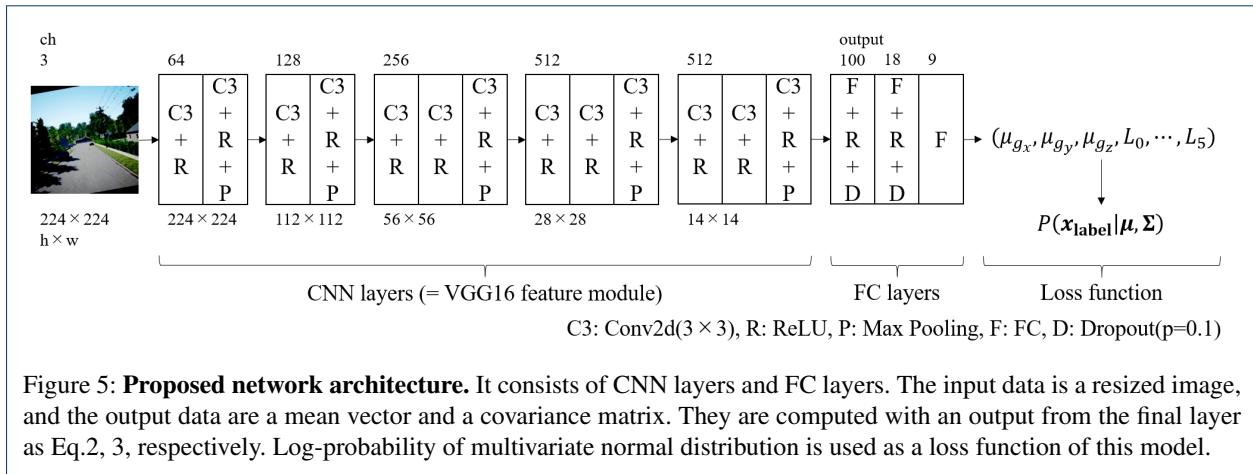
Figure 4: **Example of a transformed image.** An image is randomly rotated according to  $\Delta\phi$ . This example shows an image when  $\Delta\phi = 10$  deg. It is also resized, and is normalized.

## 2.5 Loss function

By learning the distribution of the dataset and updating the weights to maximize the probability density for the outputs, the DNN can output mean and variance. Assuming that the estimation follows a multivariate normal distribution, the probability density is computed as Eq.4.

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

where  $k$  denotes the dimensions of the variables, i.e.  $k = 3$  in the proposed method. To make the network learn the probabilistic model, it is trained maximizing  $P(\mathbf{x}_{\text{label}}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ , where  $\mathbf{x}_{\text{label}}$  ( $= \mathbf{g}_{\text{trans}}$ ) is a label of the dataset. The total probability density  $P_{\text{total}}$  of a dataset  $D_{\text{label}}$



**Figure 5: 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.2, 3, respectively. Log-probability of multivariate normal distribution is used as a loss function of this model.

is computed by multiplying as Eq.5.

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

$$\mathbf{D}_{\text{label}} = [\mathbf{x}_{\text{label}_0}, \dots, \mathbf{x}_{\text{label}_i}, \dots, \mathbf{x}_{\text{label}_n}]$$

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

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

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

$$f(\mathbf{w}) = -P_{\log_{\text{total}}} \quad (7)$$

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

## 2.6 Optimization

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

## 3 VALIDATION OF DNN

The proposed DNN was validated by comparing to other methods. The datasets used in this validation are listed in Table 1.

### 3.1 Compared methods

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

Table 1: Dataset list.

id#	Environment	#samples	Usage
1	AirSim's "Neighborhood"	10000	Training
2	"Neighborhood"	1000	Test
3	AirSim's "SoccerField"	1000	Test
4	Real world	1108	Test

### 3.1.1 MLE (ours, all)

“MLE (ours, all)” denotes the proposed method described in Section2. MLE is short for Maximum Likelihood Estimation.

### 3.1.2 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  $\beta$  in Eq.8 expresses uncertainty of the prediction, samples with small variance are selected with a threshold  $TH_\beta$ . In this validation, the threshold is set as  $TH_\beta = \frac{1}{n} \sum_{i=0}^n \beta_i$ , where  $n$  is the number of samples in the testing dataset.

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

### 3.1.3 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. Fig.6 shows the network architecture.

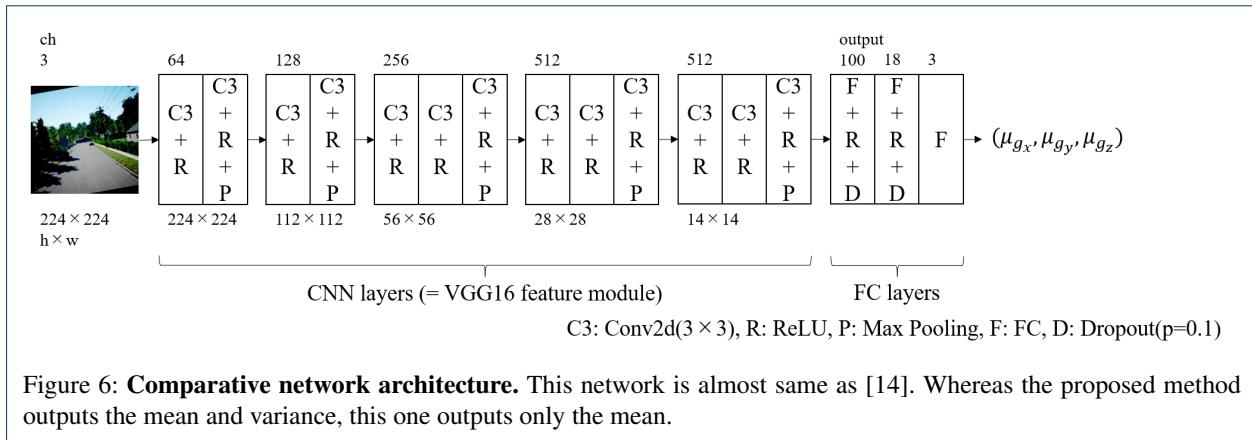


Table 2: Loss of MLE (ours) after 200 epochs.

	Dataset #			
	#1	#2	#3	#4
Loss [m/s <sup>2</sup> ]	-6.4991	-5.7436	-2.9386	-2.7129

### 3.1.4 Regression w/o L2 normalization

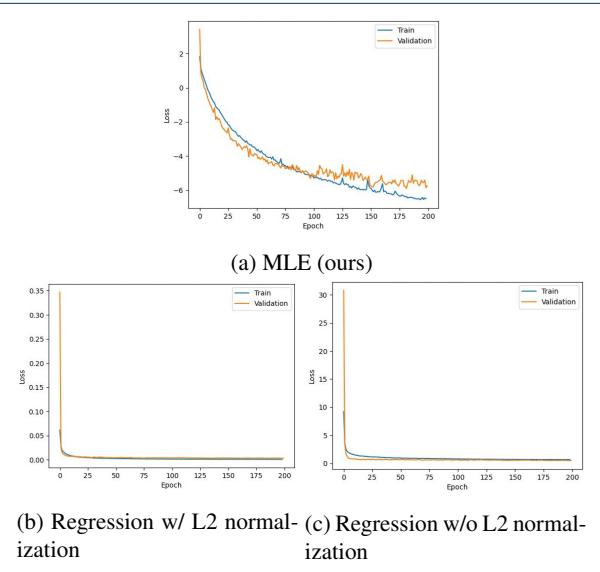
“Regression w/o L2 normalization” denotes the regression network without L2 normalization. It is required to learn not only the direction, but also norm of the gravity vector, i.e. approximately  $9.8 \text{ m/s}^2$ , in order to minimize the loss. This information is not required to estimate attitude  $\phi, \theta$ .

## 3.2 Training and validation

The network was trained with 10000 samples (#1) with a batch size of 200 samples for 200 epochs. Another 1000 samples (#2) were used for test. They were collected in “Neighborhood” of AirSim. The training dataset and the test dataset were not mixed.

Loss values during training are plotted in Fig.7. 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 2, 3 and 4 respectively show the loss values after 200 epoch training. It is noted that gradient was not computed with the test datasets. It is also noted a loss function of the MLE model and one of the regression models are difference.

Another 1000 samples (#3) were also collected in “SoccerField” of AirSim as data in an unknown environment. 1108 real data samples (#4) were also collected with the sensors in Fig.3. Note that these samples are just for test, thus the DNN was not trained with them. The loss values of these samples are larger than ones of the known environment in which the network was trained.



**Figure 7: 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.

## 3.3 Attitude estimation

Roll  $\phi$  and pitch  $\theta$  of the camera pose in the gravitational coordinate are estimated by using  $\mu$ .

$$\phi = \tan^{-1} \frac{\mu_{gy}}{\mu_{gz}}, \quad \theta = \tan^{-1} \frac{-\mu_{gx}}{\sqrt{\mu_{gy}^2 + \mu_{gz}^2}} \quad (9)$$

Mean absolute error (MAE) of the estimation of the test dataset is shown in Table 5. Variance of the estimated attitude error is shown in Table 6. Both of the error and the variance of “MLE (ours, selected)” are smaller than the others. 715 samples which has  $\beta < \text{TH}_\beta = 0.00008814 \text{ m}^3/\text{s}^6$

Table 3: Loss of Regression w/ L2 normalization after 200 epochs.

	Dataset #			
	#1	#2	#3	#4
Loss [m <sup>2</sup> /s <sup>4</sup> ]	0.0011	0.0031	0.0084	0.0055

Table 4: Loss of Regression w/o L2 normalization after 200 epochs.

	Dataset #			
	#1	#2	#3	#4
Loss [m <sup>2</sup> /s <sup>4</sup> ]	0.6588	0.4916	1.0951	0.8284

were selected from 1000 test samples (#2) with “MLE (ours, selected)”.

Comparing “MLE (ours, all)” and “MLE (ours, selected)”, filtering by  $\text{TH}_\beta$  is found valid, which means the network expresses the uncertainty by outputting the covariance matrix. In order to see this, the samples are sorted in Fig.8. In Fig.8(a), top 50 samples with largest estimation error with “MLE (ours)” are shown in order. In Fig.8(b), top 50 samples with the largest  $\beta$  with “MLE (ours)” are shown in order. 21 samples of the top 50 samples are mutual of both groups. In Fig.8(a), most of the sample images with large error are covered by obstacles, and they are dark with much less landscape information. There is no way to detect them by the regression model. Correlation between error and  $\beta$  are not complete, but many samples with large error were detected by sorting samples with  $\beta$ . A good example with large  $\beta$  and one with small  $\beta$  are shown in Fig.9, respectively. Obviously, the sample in Fig.9(a) does not have enough landscape information to estimate the gravity direction, and the proposed network expresses the uncertainty with large  $\beta$ .

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.

A dataset of the unknown environment “SoccerField” (#3) and one of real world (#4) were also used for validation. Note that the neural networks were not trained in these environments. MAE and variance of the error are shown in Table 7, 8, 9 and 10, respectively. Filtering by the threshold  $\text{TH}_\beta$  made error smaller even in the unknown environments. However, the errors and the variances of errors are larger than the known environment (Table 5 and 6). To reduce difference of results between in known environments and in unknown ones, wider variety of datasets are needed for training.

## 4 EKF-BASED SELF-ATTITUDE ESTIMATION WITH DNN

The proposed method integrates angular velocities from a gyroscope and the DNN outputs in the EKF. The proposed

Table 5: MAE of estimated attitude in known environment (#2).

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

Table 6: Variance of estimated attitude error in known environment (#2).

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

EKF architecture is shown in Fig.10. The state vector  $\boldsymbol{x}$  of the proposed Kalman filter consists of roll and pitch of the robot. Both of the vector  $\boldsymbol{x}$  and the covariance matrix  $\boldsymbol{P}$  are computed in a prediction process and in an update process. The prediction process is computed by integrating angular velocity from a gyroscope. The update process is computed by observing the output of the DNN.

$$\boldsymbol{x} = (\phi \quad \theta)^T \quad (10)$$

Here,  $k$  denotes time step,  $S_\phi$ ,  $C_\phi$ ,  $T_\phi$  are short for  $\sin \phi$ ,  $\cos \phi$ ,  $\tan \phi$ , respectively in the following sections.

### 4.1 Prediction process

The state vector  $\boldsymbol{x}$  and the covariance matrix  $\boldsymbol{P}$  are respectively computed as following.

$$\begin{aligned} \bar{\boldsymbol{x}}_k &= f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) = \boldsymbol{x}_{k-1} + \boldsymbol{Rot}_{(\boldsymbol{x}_{k-1})}^{\text{Rpy}} \boldsymbol{u}_{k-1} \\ \boldsymbol{u}_{k-1} &= \boldsymbol{\omega}_{k-1} \Delta t = \begin{pmatrix} \omega_{x_{k-1}} \Delta t \\ \omega_{y_{k-1}} \Delta t \\ \omega_{z_{k-1}} \Delta t \end{pmatrix} \\ \boldsymbol{Rot}_{(\boldsymbol{x}_{k-1})}^{\text{Rpy}} &= \begin{pmatrix} 1 & S_{\phi_{k-1}} T_{\theta_{k-1}} & C_{\phi_{k-1}} T_{\theta_{k-1}} \\ 0 & C_{\phi_{k-1}} & -S_{\phi_{k-1}} \end{pmatrix} \end{aligned} \quad (11)$$

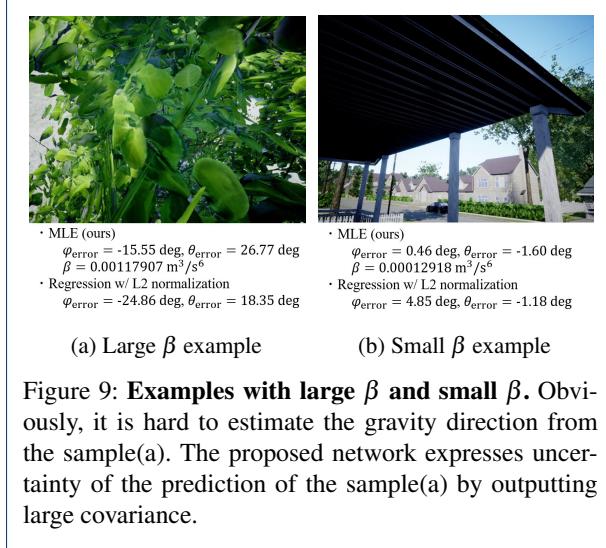
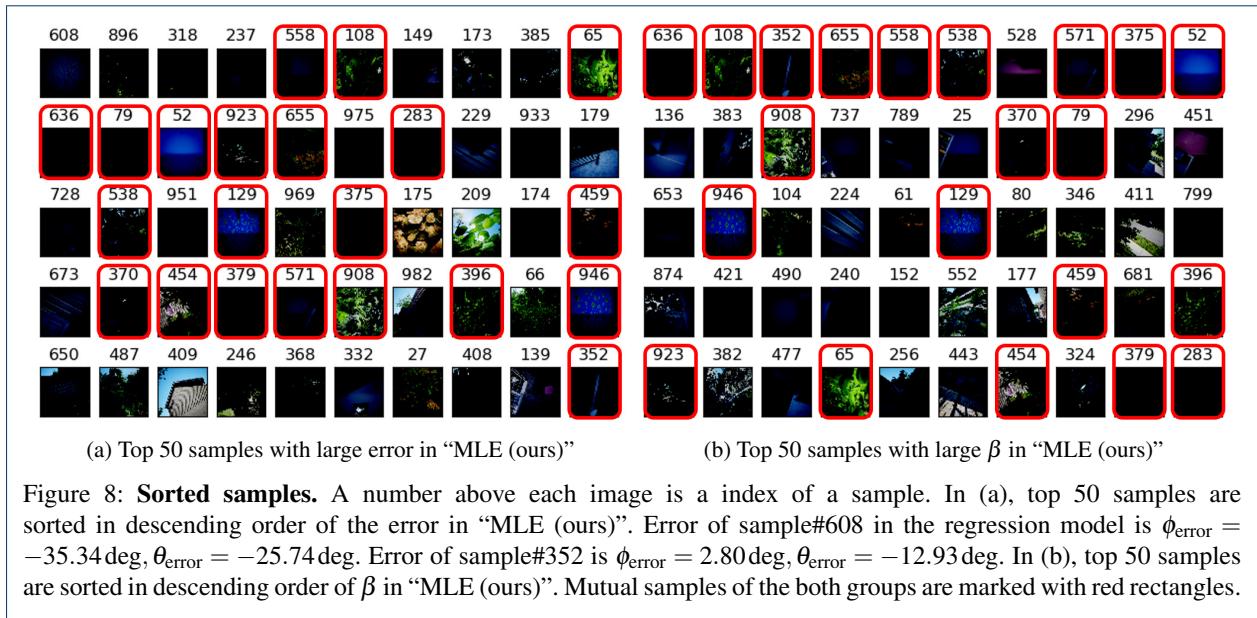
where  $f$  is a state transition model,  $\boldsymbol{u}$  denotes a control vector, and  $\boldsymbol{Rot}^{\text{Rpy}}$  denotes a rotation matrix for angular velocities.

$$\bar{\boldsymbol{P}}_k = \boldsymbol{J}_{\boldsymbol{f}_{k-1}} \boldsymbol{P}_{k-1} \boldsymbol{J}_{\boldsymbol{f}_{k-1}}^T + \boldsymbol{Q}_{k-1}, \quad \boldsymbol{J}_{\boldsymbol{f}_{k-1}} = \left. \frac{\partial f}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}} \quad (12)$$

where  $\boldsymbol{J}_{\boldsymbol{f}}$  denotes  $f$  Jacobian, and  $\boldsymbol{Q}$  denotes a covariance matrix of the process noise.

### 4.2 Update process

Outputs of the DNN with a large variance value  $\beta$  are rejected. A threshold  $\text{TH}_\beta$  is set for judging  $\beta$ , and only out-



puts with  $\beta < \text{TH}_\beta$  are observed in the EKF. The observation vector is  $\mathbf{z}$ .

$$\mathbf{z} = \boldsymbol{\mu} \quad (13)$$

where  $\boldsymbol{\mu}$  denotes a mean vector of gravity which is output from the DNN. The observation model is  $h$ .

$$h_{(\mathbf{x}_k)} = \mathbf{Rot}_{(\mathbf{x}_k)}^{\text{xyz}} \mathbf{g}_{\text{world}}, \quad \mathbf{g}_{\text{world}} = \begin{pmatrix} 0 \\ 0 \\ g_{\text{world}} \end{pmatrix} \quad (14)$$

Table 7: MAE of estimated attitude in unknown environment (#3).

	Roll [deg]	Pitch [deg]
MLE (ours, all)	3.120	4.062
MLE (ours, selected)	<b>2.069</b>	<b>2.549</b>
Regression w/ L2 normalization	3.796	4.386
Regression w/o L2 normalization	3.744	4.695

Table 8: Variance of estimated attitude error in unknown environment (#3).

	Roll [deg <sup>2</sup> ]	Pitch [deg <sup>2</sup> ]
MLE (ours, all)	31.170	36.967
MLE (ours, selected)	<b>9.134</b>	<b>12.932</b>
Regression w/ L2 normalization	35.358	45.428
Regression w/o L2 normalization	36.460	55.347

$$\mathbf{Rot}_{(\mathbf{x}_k)}^{\text{xyz}} = \begin{pmatrix} C_{\theta_k} C_{\psi_k} & C_{\theta_k} S_{\psi_k} & -S_{\theta_k} \\ S_{\phi_k} S_{\theta_k} C_{\psi_k} - C_{\phi_k} S_{\psi_k} & S_{\phi_k} S_{\theta_k} S_{\psi_k} + C_{\phi_k} C_{\psi_k} & S_{\phi_k} C_{\theta_k} \\ C_{\phi_k} S_{\theta_k} C_{\psi_k} + S_{\phi_k} S_{\psi_k} & C_{\phi_k} S_{\theta_k} S_{\psi_k} - S_{\phi_k} C_{\psi_k} & C_{\phi_k} C_{\theta_k} \end{pmatrix}$$

where  $\mathbf{g}_{\text{world}}$  denotes a gravity vector in the world frame i.e.  $g_{\text{world}} \doteq 9.8 \text{ m/s}^2$ ,  $\mathbf{Rot}^{\text{xyz}}$  denotes a rotation matrix for a vector. The covariance matrix of the process noise is  $\mathbf{R}$ .

$$\mathbf{R} = \begin{pmatrix} \gamma \Sigma_{0,0} & \Sigma_{0,1} & \Sigma_{0,2} \\ \Sigma_{1,0} & \gamma \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,0} & \Sigma_{2,1} & \gamma \Sigma_{2,2} \end{pmatrix} \quad (15)$$

where  $\gamma$  denotes a hyperparameter for adjusting variance. The state vector  $\mathbf{x}$  and the covariance matrix  $\mathbf{P}$  are respec-

Table 9: MAE of estimated attitude in real world (#4).

	Roll [deg]	Pitch [deg]
MLE (ours, all)	2.648	4.361
MLE (ours, selected)	<b>2.199</b>	<b>3.755</b>
Regression w/ L2 normalization	2.812	4.665
Regression w/o L2 normalization	3.092	5.371

Table 10: Variance of estimated attitude error in real world (#4).

	Roll [deg <sup>2</sup> ]	Pitch [deg <sup>2</sup> ]
MLE (ours, all)	20.548	20.585
MLE (ours, selected)	<b>9.356</b>	<b>13.971</b>
Regression w/ L2 normalization	20.976	20.982
Regression w/o L2 normalization	25.996	24.160

tively computed as following.

$$\hat{x}_k = x_k + \mathbf{K}_k(z_k - h_{(x_k)}), \quad \hat{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{J}_{hk}) \mathbf{P}_k$$

$$\mathbf{J}_{hk} = \frac{\partial h}{\partial x} \Big|_{x_k}, \quad \mathbf{K}_k = \mathbf{P}_k \mathbf{J}_{hk}^T (\mathbf{J}_{hk} \mathbf{P}_k \mathbf{J}_{hk}^T + \mathbf{R}_k)^{-1} \quad (16)$$

where  $\mathbf{J}_h$  denotes  $h$  Jacobian,  $\mathbf{K}$  denotes a gain matrix, and  $\mathbf{I}$  denotes an identity matrix.

## 5 VALIDATION OF EKF

The proposed system was validated in real-time. Since ground truth is available in the simulator, we used synthetic flight data of a drone.

### 5.1 Compared methods

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

#### 5.1.1 Gyro

“Gyro” denotes an estimation method integrating angular velocity from a gyroscope.

#### 5.1.2 Gyro+Acc

“Gyro+Acc” denotes an EKF-based estimation method integrating angular velocity and linear acceleration from IMU.

#### 5.1.3 Gyro+NDT

“Gyro+NDT” denotes NDT SLAM[4] using 32 layers of LiDAR. Angular velocity from a gyroscope, linear velocity of ground truth and NDT output are integrated in the EKF. Note that linear velocity of ground truth is available because the environment is a simulator.

#### 5.1.4 Gyro+Regression

“Gyro+Regression” denotes an EKF-based estimation method integrating angular velocity from a gyroscope and gravity vectors from the regression network (Fig.6). All outputs from the network are integrated in the EKF.

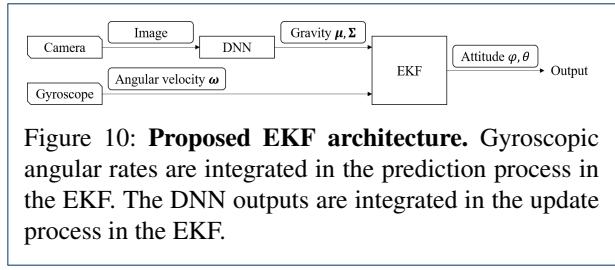


Figure 10: **Proposed EKF architecture.** Gyroscopic angular rates are integrated in the prediction process in the EKF. The DNN outputs are integrated in the update process in the EKF.

### 5.1.5 Gyro+MLE (ours)

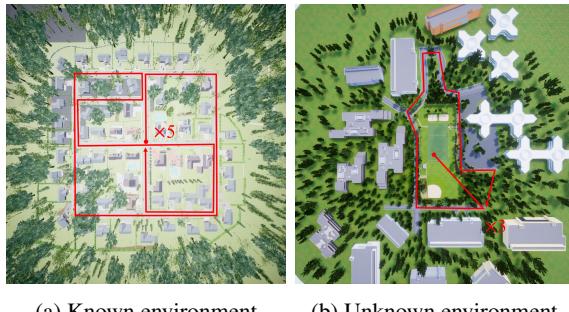
“Gyro+MLE (ours)” denotes the proposed method described in Section 4. The hyperparameters were set as  $TH_\beta = 8 \times 10^{-5}$  and  $\gamma = 1 \times 10^4$ . They were empirically determined based on the result in Section 3.

## 5.2 Experimental conditions

Flight data of a drone were recorded in “Neighborhood” and “SoccerField” of AirSim. “Neighborhood” is the same environment where the DNN was trained. An IMU and a camera are installed on a drone in the simulator. A LiDAR with 32 layers is also used for “Gyro+NDT”. The sampling frequency of the IMU, the camera and LiDAR are approximately 100 Hz, 12 Hz and 20 Hz, respectively. The camera’s horizontal FOV is 70 deg. Virtual noise was added to the IMU’s 6-axis data. It was randomly added following a normal distribution with a mean of 0 rad/s, 0 m/s<sup>2</sup> and a standard deviation of 0.1 rad/s, 0.1 m/s<sup>2</sup>, respectively. The flight courses of “Neighborhood” and “SoccerField” are shown in Fig.11. A computer which has i7-6700 CPU and GTX1080 GPU with 16 GB memory was used for the estimation. The DNN inference computation takes around 0.01-0.02 seconds with the computer. The proposed method is not only for UAVs, but it needs to be noted that this computer may be over-performance for real UAVs. Therefore, testing with a general UAV’s computer specification should be necessary when the method is used on real UAVs, and it is not the focus of this study.

## 5.3 Experimental results

The estimated attitudes are plotted in Fig.12(a). Table 11 shows MAE of the estimated attitudes in “Neighborhood”. MAE of “Gyro+MLE (ours)” is smaller than ones of the other methods. “Gyro” had large accumulative error. That is natural because noise was added and the method does not have any other observation. “Gyro+Acc” does not have accumulative error. However it has error constantly, since the acceleration values of the sensor contain own acceleration of the robot. “Gyro+NDT” accumulated error slower than “Gyro” did by using LiDAR, but it could not remove the accumulation. “Gyro+Regression” and “Gyro+MLE (ours)” corrected the accumulative error by observing the estimated gravity. Comparing “Gyro+Regression” and “Gyro+MLE (ours)”, filtering out the DNN outputs with



(a) Known environment (b) Unknown environment

**Figure 11: Flight courses.** In (a), a course in “Neighborhood” which is a known environment for the DNN is shown. The drone flew for 5 rounds. It took about 22 minutes. In (b), a course in “SoccerField” which is an unknown environment for the DNN is shown. The drone flew for 3 rounds. It took about 6 minutes.

Table 11: MAE of estimated attitude during flight in known environment.

	Roll [deg]	Pitch [deg]
Gyro	14.524	12.562
Gyro+Acc	2.920	3.380
Gyro+NDT	7.456	6.516
Gyro+Regression	3.187	1.876
Gyro+MLE (ours)	<b>2.869</b>	<b>1.821</b>

large  $\beta$  is found valid. With “Gyro+MLE (ours)”, 31 % of the outputs is rejected by the threshold in the flight.

MAE of the estimation in “SoccerField” is shown in Table 12. “Gyro+Regression” and “Gyro+MLE (ours)” suppressed accumulative error even in the environment where the DNN was not trained. MAE of “Gyro+MLE (ours)” is smaller than ones of the other methods. With “Gyro+MLE (ours)”, more outputs (58 %) from the DNN are filtered out by the threshold  $TH_\beta$  in the unknown environment (“SoccerField”) than in the known environment (“Neighborhood”). This can avoid observing outputs with high uncertainty. However, it leads chances of correcting error less. In other words, the proposed method can not correct error when the large variance is continuing. To compensate this decrease of the chances, the proposed method actually can integrates other observations such as IMU’s acceleration, SLAM, and so on in a future work. Whereas, in this experiment, the proposed method integrates only angular rate and the DNN outputs in order to make the validation simple.

## 6 CONCLUSIONS AND FUTURE WORK

EKF-based self-attitude estimation with DNN learning landscape information was proposed. The DNN estimates direction of gravity from a single shot image. The network outputs not only the gravity vector, but also a covariance matrix. Training was done with synthetic data of AirSim,

Table 12: MAE of estimated attitude during flight in unknown environment.

	Roll [deg]	Pitch [deg]
Gyro	6.424	4.679
Gyro+Acc	3.155	3.091
Gyro+NDT	4.067	2.646
Gyro+Regression	3.248	1.984
Gyro+MLE (ours)	<b>2.869</b>	<b>1.785</b>

and validation was done with both of the synthetic data and real sensors’ data. In the validation, many samples with large error are filtered out by judging variance values. It means the proposed network expresses uncertainty of the prediction by outputting covariance matrices. The outputs from the DNN and angular velocity from a gyroscope are integrated in the EKF. The covariance matrix is used adjusting process noise. Moreover, outputs with too large variance are filtered out by a threshold. This EKF-based proposed method was validated with flight data of a drone in AirSim environments. In the experiments, the proposed method suppressed accumulative error by using the DNN.

In a future work, wider variety of datasets including real data are needed for the DNN to close the gap between the results in known environments and in unknown environments. Simulator data can be used for pre-training of the DNN before fine tuning with the real data. Only the roll augmentation is applied in this paper because it is easy and simple, but pitch augmentation by the homography transformation should also be valid for fine-tuning with the less real samples. Using other sensors or combining other methods is another future work. The experiments in this paper were done only with the daytime condition, but real robots should work in day and night. Therefor the future work needs to compensate the camera’s weak point for night operations.

### Availability of data and materials

- Dataset consisting of images and their corresponding gravity vectors.  
[https://github.com/ozakiryota/dataset\\_image\\_to\\_gravity](https://github.com/ozakiryota/dataset_image_to_gravity)
- Source code for deep learning. It is implemented using Python and PyTorch API.  
[https://github.com/ozakiryota/image\\_to\\_gravity](https://github.com/ozakiryota/image_to_gravity)
- Source code for EKF framework. It is implemented using C++ and ROS API.  
[https://github.com/ozakiryota/dnn\\_attitude\\_estimation](https://github.com/ozakiryota/dnn_attitude_estimation)

### Competing interests

The authors declare that they have no competing interests.

### Funding

This study was supported by the New Energy and Industrial Technology Development Organization (NEDO).

### Author’s contributions

RO proposed the method described in this paper, implemented all the programing, conducted all the experiments, and drafted the manuscript. YK provided the inspiration for this study, provided advice, and checked and corrected the manuscript. Both authors read and approved the final manuscript.

### Acknowledgements

The authors are thankful for the generous support from the New Energy and Industrial Technology Development Organization (NEDO) for this study. This study was conducted under "Autonomous Robot Research Cluster" at Meiji University.

### References

1. Vaganay, J., Aldon, M.J., Fournier, A.: Mobile robot attitude estimation by fusion of inertial data. In: Proceedings of 1993 IEEE International Conference on Robotics and Automation (ICRA), pp. 277–282 (1993)
2. Thrun, S., Burgard, W., Fox, D.: In: Probabilistic Robotics, pp. 309–336. The MIT Press (2005)
3. Rusinkiewicz, S., Levoy, M.: Efficient variants of the icp algorithm. In: Proceedings of Third International Conference on 3-D Digital Imaging and Modeling, pp. 145–152 (2001)
4. Biber, P., er, W.S.: The normal distributions transform: A new approach to laser scan matching. In: Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2003)
5. Engel, J., Stueckler, J., Cremers, D.: Lsd-slam: Large-scale direct monocular slam. In: Proceedings of European Conference on Computer Vision (ECCV), pp. 834–849 (2014)
6. Mur-Artal, R., Montiel, J.M.M., Tardós, J.D.: Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics* **31**(5), 1147–1163 (2015)
7. Quddus, M.A., Ochieng, W.Y., Noland, R.B.: Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies* **15**(5), 312–328 (2007)
8. Kim, P., Coltin, B., Kim, H.J.: Linear rgbd slam for planar environments. In: Proceedings of European Conference on Computer Vision (ECCV), pp. 333–348 (2018)
9. Hwangbo, M., Kanade, T.: Visual-inertial uav attitude estimation using urban scene regularities. In: Proceedings of 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 2451–2458 (2011)
10. do Monte Lima, J.P.S., Uchiyama, H., Taniguchi, R.I.: End-to-end learning framework for imu-based 6-dof odometry. *Sensors* **2019** **19**(17), 3777 (2019)
11. Al-Sharman, M.K., Zweiri, Y., Jaradat, M.A.K., Al-Husari, R., Gan, D., Seneviratne, L.D.: Deep-learning-based neural network training for state estimation enhancement: Application to attitude estimation. *IEEE Transactions on Instrumentation and Measurement* **69**(1), 24–34 (2020)
12. Mérida-Floriano, M., Caballero, F., Acedo, D., García-Morales, D., Casares, F., Merino, L.: Bioinspired direct visual estimation of attitude rates with very low resolution images using deep networks. In: Proceedings of 2019 IEEE International Conference on Robotics and Automation (ICRA), pp. 5672–5678 (2019)
13. Shah, S., DeyChris, D., Kapoor, L.: Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *Field and Service Robotics* **5**, 621–635 (2017)
14. Ellingson, G., Wingate, D., McLain, T.: Deep visual gravity vector detection for unmanned aircraft attitude estimation. In: Proceedings of 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017)
15. Kalman, R.E.: A new approach to linear fi Itering and prediction problems. *Journal of Basic Engineering* **82**, 35–45 (1960)
16. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: arXiv Preprint, pp. 1409–1556 (2014)
17. Deng, J., Dong, W., Socher, R., Li, K.L., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proceedings of 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 248–255 (2009)
18. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of ICML 2010, pp. 807–814 (2010)
19. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* **15**(1), 1929–1958 (2014)
20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the 3rd International Conference for Learning Representations (ICLR) (2015)

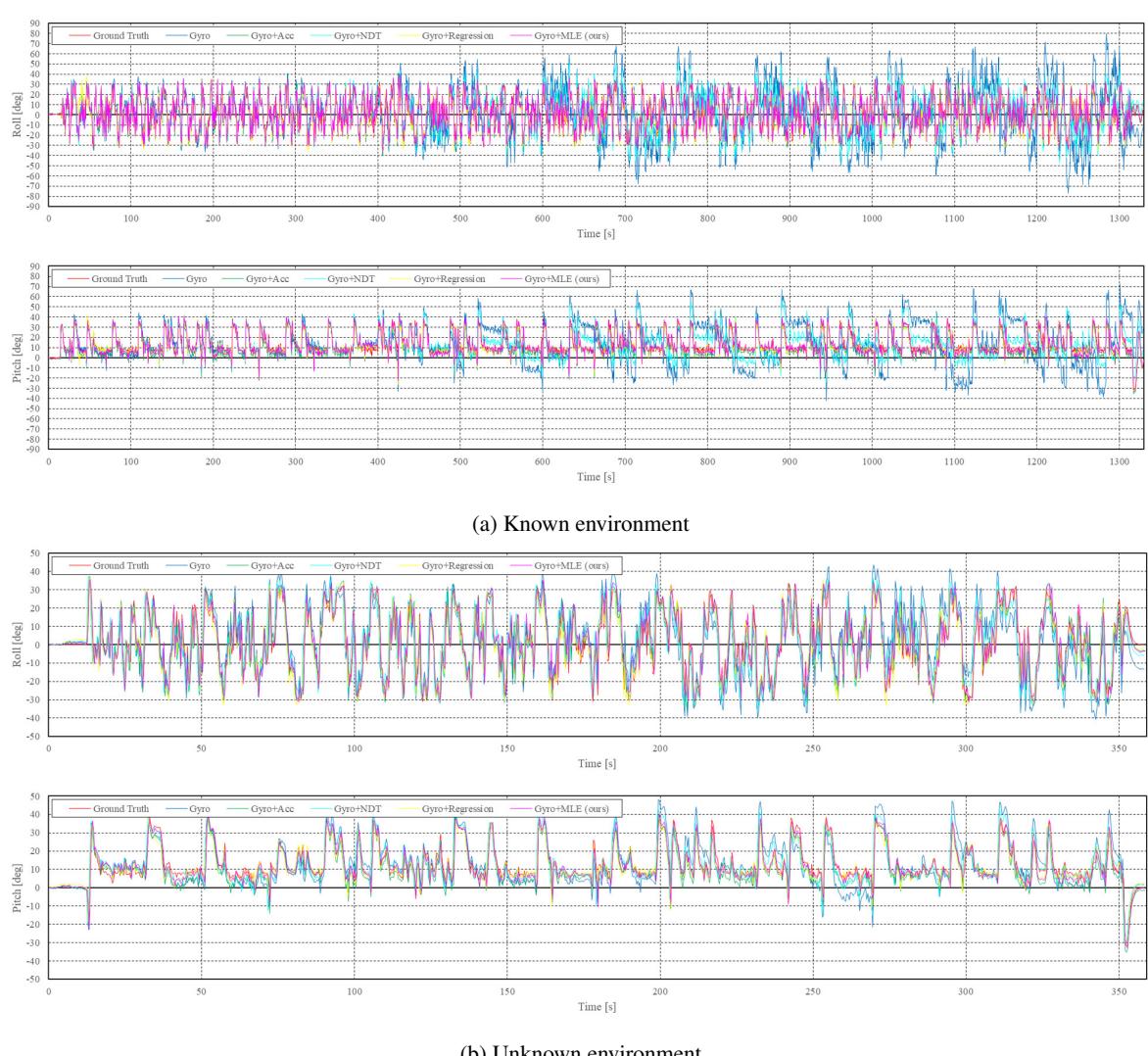


Figure 12: **Real-time attitude plotting.** In (a), the result of the flight in “Neighborhood” is shown. In (b), the result of the flight in “SoccerField” is shown.

# EKF-based real-time self-attitude estimation with camera DNN learning landscape regularities

Ryota Ozaki<sup>1</sup> and Yoji Kuroda<sup>1</sup>

**Abstract**—This paper presents an EKF (extended Kalman filter) based real-time self-attitude estimation method with a camera DNN (deep neural network) learning landscape regularities. The proposed DNN infers the gravity direction from a single shot image. It outputs the gravity direction as a mean vector and a covariance matrix in order to express uncertainty of the inference. It is pre-trained with datasets collected in a simulator. Fine-tuning with datasets collected with real sensors is carried out after the pre-training. Data augmentation is processed during the training in order to provide higher general versatility. The proposed method integrates angular rates from a gyroscope and the DNN’s outputs in an EKF. The covariance matrix output from the DNN is used as process noise of the EKF. Moreover, inferences with too large variance are filtered out before processing the integration in the EKF. Static validations are performed to show the DNN can infer the gravity direction with uncertainty expression. Dynamic validations are performed to show the DNN can be used in real-time estimation. Some conventional methods are implemented for comparison.

**Index Terms**—Localization, Visual Learning, Computer Vision for Automation

## I. INTRODUCTION

ESTIMATING the 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 accelerometers and gyroscopes. However, mobile robots have their own acceleration. Moreover, on-road robots also receive pulses from the ground, and UAVs suffer from vibration of their multi-rotor. These need to be filtered out from the accelerometer. On the other hand, integration of gyroscopic angular rate has problems 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 LiDAR odometry, VO (visual odometry) and SLAM (simultaneous localization and mapping) [2] have been proposed. LiDAR-based methods register point clouds by ICP [3], NDT [4], and so on. Visual methods often track features in image

Manuscript received: October 13, 2020; Revised January 7, 2021; Accepted February 8, 2021.

This paper was recommended for publication by Editor S. Behnke upon evaluation of the Associate Editor and Reviewers’ comments. This work was supported by New Energy and Industrial Technology Development Organization (NEDO).

<sup>1</sup>The authors are with Graduate School of Science and Technology, Meiji University, Kanagawa, 214-8571, Japan ce192021@meiji.ac.jp; ykuroda@meiji.ac.jp

Digital Object Identifier (DOI): see top of this page.

sequences [5], [6]. However, these odometry methods and SLAMs often contain 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 often used [7]. These methods correct the error by matching the prior information against the sensor data. However, they work only in environments where maps are available. Moreover, creating a map is time-consuming, and update is also required. Some methods [8], [9] estimate the attitude under Manhattan world assumption. They assume that planes or edges in the environment are orthogonal to each other. It helps achieving drift-free estimation. However, it is difficult for this kind of methods to avoid being affected by objects which do not satisfy the assumption.

Deep learning has been used for attitude estimation in recent years. In [10], IMU-based odometry by end-to-end learning has been proposed. In [11], a deep neural network identifies the measurement 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, context, 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 on time sequence since only a single shot image is used for every estimation. It helps suppressing drift, noise, and accumulative error. This method is similar to our DNN. However, this method contains some problems. It cannot express uncertainty of the inference; for instance, the network outputs estimation even when the camera is all covered by obstacles, when less features are captured, and so on. These outputs with large error worsen estimation when it is used in filter functions such as Kalman filter [15]. Therefore, they should be detected and be rejected before the integration.

To address these issues above, we presented a DNN inferring the gravity direction as mean and variance in order to express the uncertainty of the inference [17]. The validations in the paper shows the DNN can filter out inferences with large error by judging their variance values. However, the method can not output any estimation while the large variance is continuing, which means only static estimation is considered there. Another problem is that the DNN is tested on only simulator data in the paper. To use the DNN with real sensors for real-time estimation, this paper presents an EKF-based method which integrates a gyroscope and the DNN

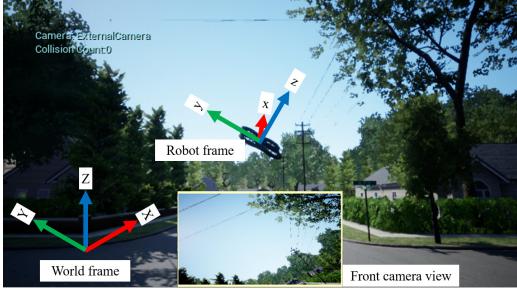


Fig. 1: Screenshot of AirSim with coordinate description. An IMU and a camera are equipped to the drone in the simulator. The purpose of this work is estimating attitude (roll and pitch) of the robot frame.

fine-tuned with real data. By integrating a gyroscope, the method can estimate the attitude with higher frequency even when the DNN can not infer the gravity direction. The data augmentation method is also updated. It is more important in this paper because collecting real data is time-consuming. The updates from the previous study [17] are summarized here:

- Mirroring, rotation, and the homography transformation are applied to the image for augmenting data more, while the previous work applies only rotation.
- Both of synthetic and real data are applied to this paper's study, while the previous study was validated on only simulator data.
- Fine-tuning after pre-training is processed to applying the network to real data efficiently.
- The DNN inference is integrated in the EKF for real-time estimation, while the previous study evaluated only the DNN outputs not in real-time. The inferred covariance matrix is used for adjusting process noise of the EKF and for filtering out large variance.

The datasets and the source code used in this paper have been released in open repositories (see APPENDIX).

## II. DNN ESTIMATING GRAVITY DIRECTION

The proposed method trains a DNN to learn landscape regularities for estimating a gravity vector in a robot frame. The gravity direction is expressed as mean and variance to consider the uncertainty of the inference.

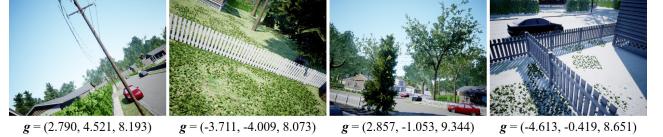
### A. Coordinate definition

A world frame is defined as a standard right-handed coordinate system. A robot frame is defined as a right-handed coordinate system which is fixed on the robot pose. They are shown in Fig.1.

### B. Dataset collection

Both of synthetic and real data are collected. The datasets consist of images and corresponded gravity vectors  $\mathbf{g}$  in the robot frame. Fig.2 shows examples of the datasets.

The synthetic datasets are collected in AirSim [13]. AirSim is a simulator for drones, cars and more, built on Unreal



(a) Synthetic data



(b) Real data

Fig. 2: Examples of datasets. The dataset consists of images and corresponded gravity vectors  $\mathbf{g}[\text{m}/\text{s}^2]$  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.

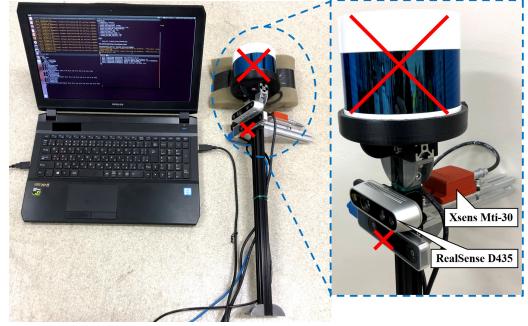


Fig. 3: Sensors installed on stick. Images and acceleration are recorded with this stick when it is still. The judge whether it is still is processed by programming. Note that depth information is not used in this study although RealSense D435 is a RGB-D camera.

Engine, which provides visually realistic graphics. An IMU and a camera are installed to a drone in the simulator in this work. The robot 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  $\phi$  and pitch  $\theta$  are limited as [-30 deg, 30 deg], respectively.

The real datasets are collected with a stick with an IMU (Xsens MTi-30) and a camera (RealSense D435) installed on a stick (Fig.3). The stick is hand-carried, and a image and a linear acceleration vector are recorded. They are saved only when the stick is shaking less than 0.001 m, 0.1 deg, and when it is at least 5 deg away from the last saved pose. The IMU is regarded as ground truth because it has enough accuracy (within 0.2 deg) in static according to the specification. Learning the static IMU is valuable because the DNN can reproduce it even in dynamic.

### C. Data preprocessing

Each input data and label data are transformed, and are augmented in each epoch of training.

- Image (input data):

Each image is flipped in 50% of probability for augmenting the dataset. After the flipping process, the homography transformation is randomly applied to the image for augmenting the pitch data. The virtual pitch variant  $\Delta\theta$  is limited as [-10 deg, 10 deg]. The transformed height  $h'$ ,  $h''$  and width  $w'$ ,  $w''$  of the image are respectively computed as following. Fig.4 may help understanding the equations.

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

where  $h$ ,  $w$  respectively denote the height and the width of the original image, and  $\text{FOV}^v (< 2\pi)$  denotes the camera's vertical field-of-view. The image is also randomly rotated for augmenting the roll data. The rotation angle  $\Delta\phi$  is limited as [-10 deg, 10 deg]. The image is resized to  $224 \times 224$ . The RGB values are normalized following  $\text{mean} = (0.5, 0.5, 0.5)$  and  $\text{std} = (0.5, 0.5, 0.5)$ . Fig.5 shows an example of the data augmentation. Note that the training time increases by about 4 times by adding the homography transformation according to our Python implementation. It does not influence the inferring time since the transformation is only applied to the training.

- Gravity vector (label data):

The gravity vector is also transformed according to the image transformation. Since the network does not need to learn the norm of the gravity, L2 normalization is also applied to the vector in order to make the training efficient.

$$\hat{\mathbf{g}} = \begin{cases} \mathbf{Rot}_{(\Delta\phi)}^x \mathbf{Rot}_{(\Delta\theta)}^y \frac{\mathbf{g}}{|\mathbf{g}|} & (\text{w/o flip}) \\ \mathbf{Rot}_{(\Delta\phi)}^x \mathbf{Rot}_{(\Delta\theta)}^y \frac{(\mathbf{g}_x, -\mathbf{g}_y, \mathbf{g}_z)^T}{|\mathbf{g}|} & (\text{w/ flip}) \end{cases} \quad (2)$$

where  $\mathbf{Rot}^x$ ,  $\mathbf{Rot}^y$  respectively denote rotation matrices along  $x$ ,  $y$  axes.

### D. Network

The proposed DNN is shown in Fig.6. It consists of CNN (convolutional neural network) layers and FC (fully connected) layers. The input to the network is the resized image, and the outputs are a mean vector of the gravity direction and a covariance matrix. Technically, the output of the final FC layer is  $(\mu_x, \mu_y, \mu_z, L_0, \dots, L_5)$ , and the mean vector  $\hat{\mu}$  and the covariance matrix  $\Sigma$  are computed as Eq.3, 4, respectively. Since

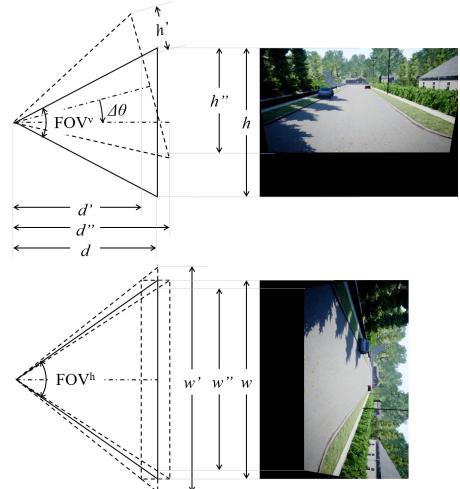


Fig. 4: Homography. The pitch data of the dataset is augmented by the homography transformation.

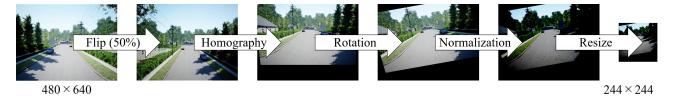


Fig. 5: Example of transformed image. Each input image is randomly flipped, transformed and rotated according to  $\Delta\theta$  and  $\Delta\phi$ . This example shows an image when  $\Delta\theta = \Delta\phi = 10$  deg. It is also resized, and is normalized.

the lower-triangular matrix  $\mathbf{L}$  is required to have positive-valued diagonal entries, an exponential function is applied to the diagonal elements.

$$\hat{\mu} = \frac{(\mu_x, \mu_y, \mu_z)^T}{|(\mu_x, \mu_y, \mu_z)^T|} \quad (3)$$

$$\Sigma = \mathbf{L}\mathbf{L}^T, \quad \mathbf{L} = \begin{pmatrix} \exp(L_0) & 0 & 0 \\ L_1 & \exp(L_2) & 0 \\ L_3 & L_4 & \exp(L_5) \end{pmatrix} \quad (4)$$

It is expected that the CNN layers lean extracting features such as edges, and the FC layers learn landscape information. Feature module of VGG16 [18] pre-trained on ImageNet [19] is adopted as the CNN layers of the proposed method. Transfer learning helps deep learning to be efficient even though the transferred network is trained on a different task [20]. All layers, except the final output layer, use the ReLU function [16] as activation function. All FC layers, except the final output layer, use the 10% Dropout [21] to avoid the overfitting problem.

### E. Loss function

By learning the distribution of the dataset and updating the weights to maximize the probability density for the outputs, the DNN can output mean and variance. Assuming that

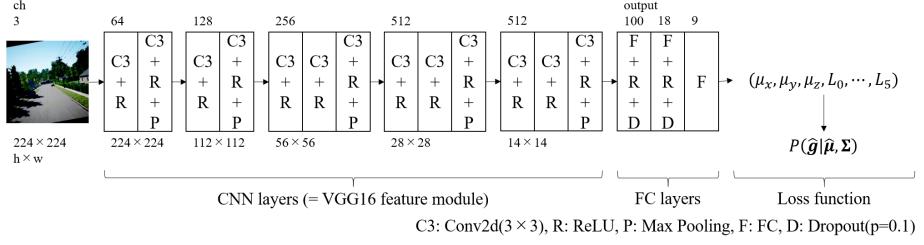


Fig. 6: Proposed network architecture. It consists of CNN layers and FC layers. The input data is a resized image, and the output data are a mean vector and a covariance matrix. They are computed with an output from the final layer as Eq.3, 4, respectively. Log-probability of multivariate normal distribution is used as a loss function of this model.

the estimation follows a multivariate normal distribution, the proposed loss function  $l$  is computed as below.

$$\begin{aligned} l(\Theta) &= -\sum_{i=0}^{\#D} \ln p(\hat{g}_i|\hat{\mu}_i, \Sigma_i), \quad d = \text{rank}(\Sigma_i) \\ p(\hat{g}_i|\hat{\mu}_i, \Sigma_i) &= \frac{\exp(-\frac{1}{2}(\hat{g}_i - \hat{\mu}_i)^T \Sigma^{-1} (\hat{g}_i - \hat{\mu}_i))}{\sqrt{(2\pi)^d |\Sigma_i|}} \end{aligned} \quad (5)$$

where  $\Theta$  denotes the parameters of the network,  $\#D$  denotes the number of samples in the dataset, and  $d$  denotes the dimensions of the variables, i.e.  $d = 3$  in the proposed method. The network minimizes the loss by updating  $\Theta$ .

#### F. Optimization

Adam (adaptive moment estimation) [22] is used to optimize the parameters. For the training with the synthetic data, the learning rates are set as  $lr_{\text{CNN}} = 0.000001$ ,  $lr_{\text{FC}} = 0.0001$ , where  $lr_{\text{CNN}}$  is a value for the CNN layers,  $lr_{\text{FC}}$  is a value for the FC layers. For the fine-tuning with the real data, they are set smaller as  $lr_{\text{CNN}} = 0.0000001$ ,  $lr_{\text{FC}} = 0.000001$ .

#### G. Uncertainty expression

In this study,  $\eta$  in Eq.6 is assumed to express uncertainty of the inference. This value  $\eta$  is used to filter out inferences with large variance.

$$\eta = \sqrt{\sigma_x^2} \times \sqrt{\sigma_y^2} \times \sqrt{\sigma_z^2}, \quad \Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z^2 \end{pmatrix} \quad (6)$$

### III. EKF-BASED REAL-TIME ESTIMATION

The outputs from the DNN are integrated with gyroscopic angular rate in EKF. The proposed EKF architecture is shown in Fig.7. The state vector  $\mathbf{x}$  of the proposed Kalman filter consists of roll  $\phi$  and pitch  $\theta$  of the robot. Both of the vector  $\mathbf{x}$  and the covariance matrix  $\mathbf{P}$  are computed in a prediction process and an update process. The prediction process is computed by integrating angular velocity from a gyroscope. The update process is computed by observing the outputs of the DNN.

$$\mathbf{x} = (\phi \quad \theta)^T \quad (7)$$

Here,  $t$  denotes the time step in the following sections.

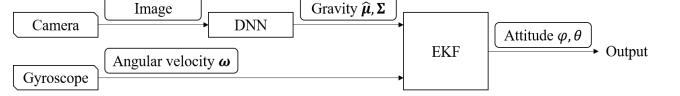


Fig. 7: Proposed EKF architecture. Gyroscopic angular rates are integrated in the prediction process in EKF. The DNN outputs are integrated in the update process in EKF.

#### A. Prediction process

The state vector  $\mathbf{x}$  and the covariance matrix  $\mathbf{P}$  are respectively computed as following.

$$\begin{aligned} \bar{\mathbf{x}}_t &= f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) = \mathbf{x}_{t-1} + \mathbf{Rot}_{(\mathbf{x}_{t-1})}^{\text{rpy}} \mathbf{u}_{t-1} \\ \mathbf{u}_{t-1} &= \boldsymbol{\omega}_{t-1} \Delta t = \begin{pmatrix} \omega_{x_{t-1}} \Delta t \\ \omega_{y_{t-1}} \Delta t \\ \omega_{z_{t-1}} \Delta t \end{pmatrix} \end{aligned} \quad (8)$$

where  $f$  is a state transition model,  $\mathbf{u}$  denotes a control vector,  $\boldsymbol{\omega}$  denotes the angular velocity measured with a gyroscope, and  $\mathbf{Rot}^{\text{rpy}}$  denotes a rotation matrix for angular velocities.

$$\bar{\mathbf{P}}_t = \mathbf{J}_{\mathbf{f}_{t-1}} \mathbf{P}_{t-1} \mathbf{J}_{\mathbf{f}_{t-1}}^T + \mathbf{Q}_{t-1}, \quad \mathbf{J}_{\mathbf{f}_{t-1}} = \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}_{t-1}, \mathbf{u}_{t-1}} \quad (9)$$

where  $\mathbf{J}_{\mathbf{f}}$  denotes  $f$  Jacobian, and  $\mathbf{Q}$  denotes a covariance matrix of the process noise.

#### B. Update process

Outputs of the DNN with large variance value  $\eta$  are rejected. A threshold  $\text{TH}_\eta$  is set for judging  $\eta$ , and only outputs with  $\eta < \text{TH}_\eta$  are observed in this EKF. The observation vector is  $\mathbf{z}$  as below.

$$\mathbf{z} = \hat{\mu} \quad (10)$$

where  $\hat{\mu}$  denotes a mean vector of the gravity which is output from the DNN. The observation model is  $h$ .

$$\mathbf{h}(\mathbf{x}_t) = \mathbf{Rot}_{(\mathbf{x}_t)}^{\text{xyz}} \frac{\mathbf{g}_{\text{world}}}{|\mathbf{g}_{\text{world}}|}, \quad \mathbf{g}_{\text{world}} = \begin{pmatrix} 0 \\ 0 \\ g_{\text{world}} \end{pmatrix} \quad (11)$$

where  $\mathbf{g}_{\text{world}}$  denotes a gravity vector in the world frame i.e.  $g_{\text{world}} = 9.8 \text{ m/s}^2$ , and  $\mathbf{Rot}^{\text{xyz}}$  denotes a rotation matrix for vectors. The covariance matrix of the process noise is  $\mathbf{R}$ .

$$\mathbf{R} = \begin{pmatrix} \xi \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \xi \sigma_y^2 & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \xi \sigma_z^2 \end{pmatrix} \quad (12)$$

TABLE I: Dataset list.

id#	Environment	#samples	Usage
1	AirSim	Neighborhood	10000 Training
2			1000 Test
3	SoccerField	1000	Test
4		Area I	1108 Fine-tuning
5	Real	Area II	539 Test

where  $\xi$  denotes a hyperparameter for adjusting the variance. The state vector  $\mathbf{x}$  and the covariance matrix  $\mathbf{P}$  are respectively computed as following.

$$\begin{aligned} \ddot{\mathbf{x}}_t &= \mathbf{x}_t + \mathbf{K}_t(\mathbf{z}_t - h_{(\mathbf{x}_t)}), \quad \check{\mathbf{P}}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{J}_{ht}) \mathbf{P}_t \\ \mathbf{J}_{ht} &= \frac{\partial h}{\partial \mathbf{x}} \Big|_{\mathbf{x}_t}, \quad \mathbf{K}_t = \mathbf{P}_t \mathbf{J}_{ht}^T (\mathbf{J}_{ht} \mathbf{P}_t \mathbf{J}_{ht}^T + \mathbf{R}_t)^{-1} \end{aligned} \quad (13)$$

where  $\mathbf{J}_{ht}$  denotes  $h$  Jacobian,  $\mathbf{K}$  denotes a gain matrix, and  $\mathbf{I}$  denotes an identity matrix.

#### IV. VALIDATION

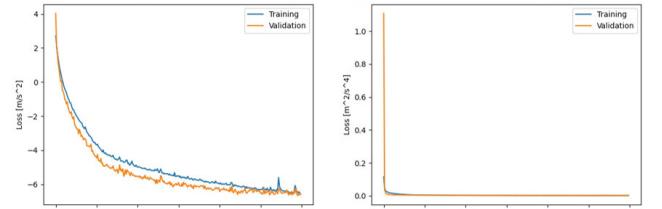
##### A. Static validation of DNN

The proposed DNN was trained with training datasets, and was evaluated with test datasets.

1) *Method list*: Definitions of methods which were used in this validation are summarized here.

- ‘MLE (ours, all)’ denotes the proposed method described in Section II. MLE is short for Maximum Likelihood Estimation.
- ‘MLE (ours, selected)’ denotes a 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 the validation of attitude estimation. It means samples with large variance are filtered out as outliers. Assuming  $\eta$  in Eq.6 expresses uncertainty of the inference, samples with small variance are selected with a threshold  $TH_\eta$ . In this validation, the threshold is set as  $TH_\eta = \frac{1}{\#D} \sum_{i=0}^{\#D} \eta_i$ , where  $\#D$  is the number of samples in the testing dataset.
- ‘Regression’ denotes a network which the final FC layer is difference from ‘MLE (ours)’. It outputs a 3D gravity vector without covariance. It is implemented base on the related work [14]. L2 normalization is applied to the final layer while ReLU is applied in [14], since ReLU outputs only positive values. MSE (mean square error) between the labels and outputs is used as a loss function.
- ‘Statistics’ denotes a method using the average of the label vectors as outputs for all samples, which means  $\sum_{i=0}^{\#D} \mathbf{g}_i$  is used for estimating attitudes of all samples. Computing the error of this method is equivalent to calculating the standard deviation of the dataset. This method is regarded as baseline in this study.

2) *Training*: The datasets used in this validation are listed in Table I. The network was trained with 10000 synthetic samples (dataset#1) with a batch size of 200 samples for 300 epochs. Another 1000 samples (dataset#2) were used for test. They were collected in ‘Neighborhood’ of AirSim. The training dataset and the test dataset were not mixed. A computer which has W-2133 CPU and Quadro GV100 GPU



(a) MLE (ours)

(b) Regression

Fig. 8: 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.

TABLE II: Loss after 300 epochs of training.

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

with 32 GB memory was used for the training. The training took around 43 hours with the computer.

The loss values during the training are plotted in Fig.8. The regression model converged much faster than the MLE model did. Table II shows the loss values after 300 epochs of training. Note that the loss function of the MLE model and one of the regression model are difference.

3) *Fine-tuning*: Fine-tuning with the real data was done after the training with the synthetic data. The network was tuned with 1108 real data samples (dataset#4) with a batch size of 200 samples for 300 epochs. Another 539 samples (dataset#5) were used for test. They were collected in the campus of Meiji University. The training dataset and the test dataset were collected in the same campus, but not in the same area.

The loss values during the fine-tuning are plotted in Fig.9. Table III shows the loss values after 300 epochs of the fine-tuning. The loss value on the real dataset became smaller by the fine-tuning. However the loss value on the test dataset is larger than one on the training dataset. To reduce the difference of the results between the training data and the test data, wider variety of datasets are needed for training.

4) *Attitude estimation*: The roll  $\phi$  and pitch  $\theta$  of the camera pose in the gravitational coordinate are estimated by using  $\hat{\mu}$ .

$$\phi = \tan^{-1} \frac{\hat{\mu}_y}{\hat{\mu}_z}, \quad \theta = \tan^{-1} \frac{-\hat{\mu}_x}{\sqrt{\hat{\mu}_y^2 + \hat{\mu}_z^2}} \quad (14)$$

The MAE (mean absolute error) of the estimation on the synthetic datasets is shown in Table IV. With ‘MLE (ours, selected)’, 795 samples which has  $\eta < TH_\eta = \frac{1}{\#D} \sum_{i=0}^{\#D} \eta_i = 0.000120 \text{ m}^3/\text{s}^6$  were selected from 1000 test samples in dataset#2. This threshold was also used for the other datasets. The number of samples selected by the threshold are shown in Table V. The MAE of the estimation on the real datasets is shown in Table VI. With the test datasets, the error of ‘MLE (ours, selected)’ is smaller than the others.

Comparing ‘MLE (ours, all)’ and ‘MLE (ours, selected)’, filtering by  $TH_\eta$  is found valid, which means the network

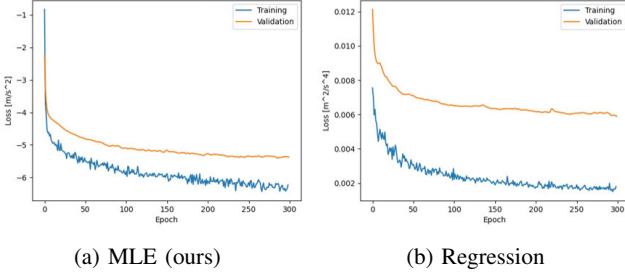


Fig. 9: Loss plotting of fine-tuning. The fine-tuning made the loss values on the real data smaller.

TABLE III: Loss after 300 epochs of fine-tuning.

	Train (#4)	Test (#5)
MLE (ours) [m/s <sup>2</sup> ]	-6.2295	-5.3756
Regression [m <sup>2</sup> /s <sup>4</sup> ]	0.0018	0.0059

expresses the uncertainty by outputting a covariance matrix. A good example with large  $\eta$  and one with small  $\eta$  are shown in Fig.10. Obviously, the sample in Fig.10a has much less landscape information to estimate the gravity direction, and the proposed network expresses the uncertainty with large  $\eta$ . There is no way to detect it by the conventional regression model.

Comparing ‘before fine-tuning’ and ‘after fine-tuning’, the fine-tuning with the real datasets makes the error smaller. The number of the samples for the fine-tuning is not large, but it worked enough. It implies the pre-training with the large synthetic dataset is valid.

Comparing with the previous study [17], the result in this paper is better. It implies the improvement of the data augmentation contributes to the accuracy. Data augmentation is especially important for real data because collecting real data is time-consuming.

### B. Validation of real-time estimation in simulator

The proposed EKF-based real-time estimation was validated on synthetic flight data of a drone since ground truth is available in the simulator.

1) *Method list:* Definitions of methods which were used in this validation are summarized here.

- ‘Gyro’ denotes an estimation method integrating angular velocity from a gyroscope.
- ‘Gyro+Acc’ denotes an EKF-based estimation method integrating angular velocity and linear acceleration from an IMU.
- ‘Gyro+NDT’ denotes NDT SLAM [4] using 32 layers of LiDAR. Angular velocity from a gyroscope, linear velocity of ground truth and the NDT output are integrated in an EKF.
- ‘Gyro+Regression’ denotes an EKF-based estimation method integrating angular velocity from a gyroscope and gravity vectors inferred by the regression network.
- ‘MLE w/o rejection’ denotes a method using the proposed DNN directly without EKF. It does not filter out any inferences in order to estimate the attitude continuously.

TABLE IV: MAE of static estimation on synthetic data.

Method	Angle [deg]	Dataset#		
		1	2	3
MLE (ours, all)	Roll	1.206	1.982	3.535
	Pitch	1.022	1.992	6.919
MLE (ours, selected)	Roll	1.014	<b>1.368</b>	<b>1.800</b>
	Pitch	<b>0.824</b>	<b>1.121</b>	<b>2.478</b>
Regression	Roll	<b>1.012</b>	1.948	2.673
	Pitch	0.852	1.902	3.693
Statistics	Roll	15.080	15.344	15.352
	Pitch	14.998	14.783	14.408

TABLE V: Number of selected samples by MAE (ours, selected).

#Selected samples (percentage [%])	Dataset#				
	1	2	3	4	5
Before fine-tuning	8388 (83.9)	795 (79.5)	368 (36.8)	672 (60.6)	225 (41.7)
After fine-tuning	-	-	-	883 (79.7)	304 (56.4)

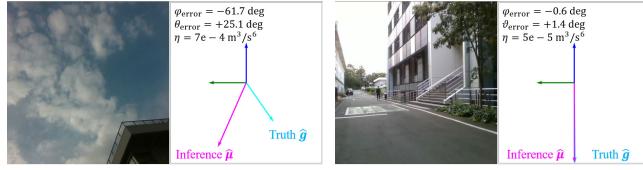
- ‘Gyro+MLE (ours)’ denotes the proposed method described in Section III. The hyperparameters were set as  $\text{TH}_\eta = 1.2 \times 10^{-4} \text{ m}^3/\text{s}^6$  and  $\xi = 5 \times 10^3$ . They were empirically determined based on the result in Section IV-A.

2) *Experimental conditions:* Flight data of a drone was recorded in ‘Neighborhood’ and ‘SoccerField’ of AirSim. The sampling frequency of the IMU and the camera are approximately 100 Hz, 12 Hz, respectively. Virtual noise was added to the IMU’s 6-axis data. It was randomly added following a normal distribution with a mean of 0 rad/s, 0 m/s<sup>2</sup> and a standard deviation of 0.1 rad/s, 0.1 m/s<sup>2</sup>, respectively. The flight courses are shown in Fig.11a, 11b. A computer which has i7-6700 CPU and GTX1080 GPU with 16 GB memory was used for the estimation. The DNN inference computation takes around 0.01 - 0.02 seconds with the computer.

3) *Experimental results:* The estimated attitudes in ‘Neighborhood’ are plotted in Fig.12. Table VII shows the MAE of the estimated attitude. The MAE of ‘Gyro+MLE (ours)’ is smaller than ones of the other methods. ‘Gyro’ had large accumulative error. That is natural because noise was added and the method does not have any other observation. ‘Gyro+Acc’ does not have accumulative error. However it has error constantly, since the acceleration values of the sensor contain own acceleration of the robot and noise. On the other hand, the proposed method can observe the gravity vector which does not contain them. ‘Gyro+NDT’ accumulated error slower than ‘Gyro’ did by using the LiDAR, but it could not remove the accumulation. ‘Gyro+Regression’ and ‘Gyro+MLE (ours)’ corrected the accumulative error by observing the estimated gravity. Comparing ‘Gyro+Regression’ and ‘Gyro+MLE (ours)’, filtering out the DNN outputs with large  $\eta$  is found valid. The error difference between them in the training environment (‘Neighborhood’) is quite small. It is considered to be because the DNNs fit the training dataset well, which led to the environment having less uncertainty to be filtered out. In the unknown environment (‘SoccerField’), ‘Gyro+MLE (ours)’ showed stronger improvement over ‘Gyro+Regression’.

TABLE VI: MAE of static estimation on real data.

Method		Angle [deg]	Dataset# 4 5		
Before fine-tuning	MLE (ours, all)	Roll	2.272	3.484	
		Pitch	4.816	5.828	
	MLE (ours, selected)	Roll	<b>1.762</b>	<b>1.951</b>	
		Pitch	<b>4.043</b>	<b>4.551</b>	
	Regression	Roll	2.217	3.140	
		Pitch	4.292	5.612	
After fine-tuning	MLE (ours, all)	Roll	1.505	2.992	
		Pitch	1.349	3.273	
	MLE (ours, selected)	Roll	<b>1.253</b>	<b>1.656</b>	
		Pitch	<b>1.114</b>	<b>2.364</b>	
	Regression	Roll	1.264	2.567	
		Pitch	1.296	3.121	
Statistics		Roll	15.803	14.125	
		Pitch	10.277	13.222	

(a) Large  $\eta$  example (b) Small  $\eta$  exampleFig. 10: Examples with  $\eta$  values.

With ‘Gyro+MLE (ours)’, about 13 % and 43 % of the inferences were rejected by the threshold  $TH_\eta$  during the flight in ‘Neighborhood’ and ‘SoccerField’, respectively. This can avoid observing outputs with high uncertainty. Especially, ‘MLE w/o rejection’ tended to output large error and large variance when the attitude angles exceeded the trained range i.e. [-30 deg, 30 deg].

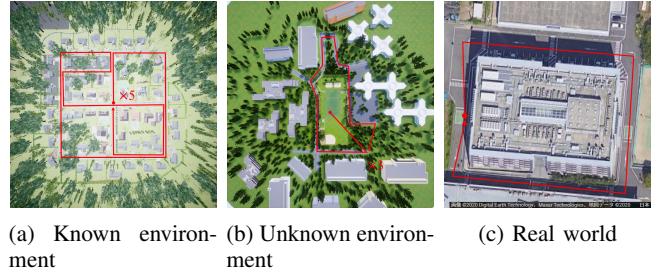
### C. Validation of real-time estimation in real world

To see the fine-tuned DNN can work in real world, two types of experiments with real sensors (Fig.3) were performed.

1) *Indoor experiment with motion capture*: The sensors were hand-carried in an indoor environment of  $4.5 \text{ m} \times 6 \text{ m}$  for about 23 minutes. Motion capture cameras (Vicon Vero v1.3X) were used for measuring the ground truth. Note that the DNN was not trained in this area.

Table VIII shows the MAE of the estimated attitude. The proposed method suppressed accumulation of error also in real world. In the flat indoor environment, the MAE given by the proposed method was almost the same as that of ‘Gyro+Acc’. The acceleration measured with the IMU is not integrated in the proposed EKF in this paper just for making the validation simple, but it actually can be integrated, and it would be more stable estimation. For reference, the error given by ‘Gyro+Acc+MLE’ was  $\phi_{\text{error}} = 2.290 \text{ deg}$ ,  $\theta_{\text{error}} = 1.618 \text{ deg}$ .

2) *Outdoor experiment*: The motion capture cameras measure the attitude accurately, but the captured area is limited. To complement that, a long distance experiment was also performed. Detailed quantitative evaluation of the accuracy was done in the previous section, thus this section is just for seeing that the proposed method also be able to work in real world.



(a) Known environment (b) Unknown environment (c) Real world

Fig. 11: Driving courses. The AirSim’s drone flew in ‘Neighborhood’ (a) for 5 rounds for about 22 minutes, and ‘SoccerField’ (b) for 3 rounds for about 6 minutes, respectively. The real sensors were carried in the campus (c) for around 5 minutes.

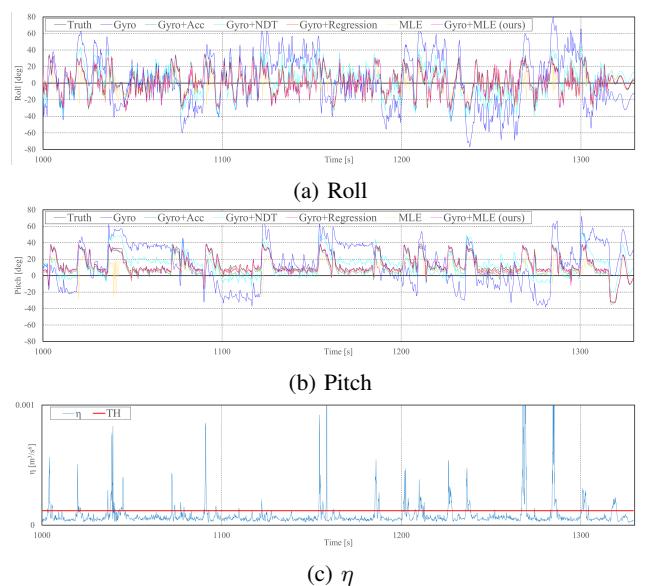


Fig. 12: Real-time plotting in ‘Neighborhood’. The graphs show the last 330 seconds of the synthetic flight. ‘MLE’ tended to output large variance  $\eta$  when the error is large.

The sensors were hand-carried for around 5 minutes in Area II (Fig.11c) where the DNN was not trained. Since the ground truth is not available while the sensors are being carried, the estimated attitude at the end of carrying was evaluated to see error accumulation. The sensors were placed on a flat floor at the start and end of the experiment as Fig.13, and the ground truth was assumed as  $\phi_{\text{gt}} = 0 \text{ deg}$ ,  $\theta_{\text{gt}} = 0 \text{ deg}$ . This evaluation method is based on the related study [14].

Table IX shows the error of the estimation at the last pose. The proposed method suppressed accumulation of error during the driving outdoor.

## V. CONCLUSIONS AND FUTURE WORK

The proposed method integrates a gyroscope and the DNN for estimating self-attitude in real-time. The proposed network estimates the gravity direction from a single shot image. The network outputs not only the gravity vector, but also a covariance matrix. It was trained with synthetic data, and was fine-

TABLE VII: MAE of dynamic estimation in simulator.

Method	Neighborhood		SoccerField	
	Roll [deg]	Pitch [deg]	Roll [deg]	Pitch [deg]
Gyro	14.524	12.562	6.424	4.679
Gyro+Acc	2.920	3.380	3.155	3.091
Gyro+NDT	7.456	6.516	4.067	2.646
Gyro+Regression	2.793	1.645	3.293	2.049
MLE w/o rejection	5.016	5.711	4.098	4.926
Gyro+MLE (ours)	<b>2.703</b>	<b>1.598</b>	<b>2.949</b>	<b>1.777</b>

TABLE VIII: MAE of dynamic estimation in mocap area.

	Roll [deg]	Pitch [deg]
Gyro	6.012	5.100
Gyro+Acc	2.509	<b>1.648</b>
Gyro+Regression	2.840	2.764
MLE w/o rejection	4.808	7.858
Gyro+MLE (ours)	<b>2.407</b>	1.902

tuned with real data. Pre-training with the large synthetic data and augmenting the data help making the learning efficient. The static experiment showed the DNN can infer the gravity direction with the uncertainty. For dynamic estimation, angular rates from a gyroscope and the DNN's outputs are integrated in the EKF. The inferred covariance matrices are used for adjusting the process noise and for filtering out infers with large variance. The dynamic experiments showed the proposed method can be used for real-time estimation.

Using multiple cameras or other sensors for estimating the attitude is our future work.

## APPENDIX

- Source code and dataset. The code is implemented using Python, C++, PyTorch API and ROS API.

[https://github.com/ozakiryota/dnn\\_attitude\\_estimation](https://github.com/ozakiryota/dnn_attitude_estimation)

## ACKNOWLEDGMENT

We 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), pp.2743–2748, 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, pp.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. Shah, D. Dey, C. Lovett and A. Kapoor, AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles, Field and Service Robotics, pp.621–635, 2018.
- [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), pp.5557–5563, 2017.
- [15] R. E. Kalman, A new approach to linear filtering and prediction problems, Journal of Basic Engineering, Vol.82, pp.35–45, 1960.
- [16] V. Nair and G. E. Hinton, Rectified linear units improve restricted boltzmann machines, Proceedings of ICML 2010, pp.807–814, 2010.
- [17] R. Ozaki, and Y. Kuroda, DNN-based self-attitude estimation by learning landscape information, Proceedings of 2021 IEEE/SICE International Symposium on System Integration (SII2021), pp.733–738, 2021.
- [18] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2014.
- [19] 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.
- [20] S. J. Pan and Q. Yang, A Survey on Transfer Learning, IEEE Transactions on Knowledge and Data Engineering, Vol.22, No.10, pp.1345–1359, 2010.
- [21] 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.
- [22] Diederik P. Kingma and Jimmy Ba, Adam: A method for stochastic optimization, Proceedings of the 3rd International Conference for Learning Representations (ICLR), 2015.

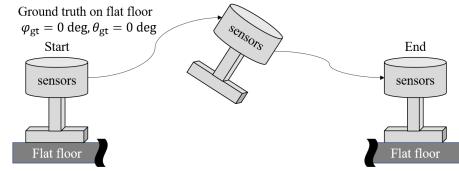


Fig. 13: Dynamic experiment. The ground truth on the flat floor is assumed to be  $\phi_{gt} = 0$  deg,  $\theta_{gt} = 0$  deg.

TABLE IX: Error of estimated attitude at last pose in outdoor.

	Roll [deg]	Pitch [deg]
Gyro+MLE (ours)	<b>+0.643</b>	<b>+1.851</b>
Gyro	+5.268	-5.047

## 建造物の壁に対する相対姿勢を用いた姿勢推定

尾崎 亮太<sup>\*1</sup>, 黒田 洋司<sup>\*2</sup>

### Pose Estimation with Relative Poses to Walls of Buildings

Ryota OZAKI<sup>\*1</sup> and Yoji KURODA<sup>\*2</sup>

<sup>\*1\*2</sup> Department of Mechanical Engineering, School of Science and Technology, Meiji University  
1-1-1 Higashimita, Kawasaki Tama-ku, Kanagawa 214-8571, Japan

This paper presents a pose estimation method which is robust to accumulative error by estimating relative poses to building walls for mobile robots in urban areas. This method exploits that most of artificial walls are built vertically. It estimates poses by not only an inertial sensor and real-time SLAM but also observations of normals from point cloud of artificial walls for estimating absolute poses in the gravity coordinate system. And those three types of poses which are estimated by each way are combined by extended Kalman filter. To evaluate the proposed method, outdoor experiments with an actual robot are performed. It shows the method keeps correcting accumulative error while the robot is driven.

**Key Words :** Pose estimation, Mobile robot

#### 1. 緒 言

移動ロボットのナビゲーションや姿勢制御を行うためには、3次元空間における時々刻々の姿勢を推定する必要がある。移動体の姿勢推定法として、内界センサを使ったデッドレコニングによる推定と、外界センサを使ったSLAMによる推定とを統合する手法が提案されている<sup>(1)</sup>。このような手法は、初期姿勢に対する相対変化を推定するため、蓄積誤差を補正することが難しい。

そこで本研究では、従来用いられる、慣性センサとSLAMの統合に加え、鉛直に建てられた建造物の壁に対する相対姿勢をワールド座標系における絶対姿勢として観測することで、誤差の蓄積を適時補正することができる姿勢推定法を提案する。なお、本手法は、一般的な建造物の壁がおおよそ鉛直に建てられていることを利用し、事前環境地図を必要としない。そして、屋外での実機走行実験によって本手法の有用性を示す。

#### 2. 建造物の壁に対する相対姿勢を用いた姿勢推定法

まず座標系を以下に定義する。

- ロボット座標系：ロボットに固定され、進行方向をx軸の正とする右手直交座標系とする。
- ワールド座標系：ロボット初期姿勢位置において

原点がロボット座標系と一致し、重力方向をz軸の負とする右手直交座標系とする。

本提案手法では、ロボット静止時にIMUによって初期姿勢を求め、それに対してデッドレコニングおよびSLAMで推定される相対姿勢変化を積算する。そして、壁面を観測した場合は、2・1, 2・2章のように推定したロボット座標系での重力ベクトルを用いて、蓄積誤差を補正する。なお、デッドレコニングで推定する姿勢を予測、SLAMで推定する姿勢を観測1、壁面に対する相対姿勢を観測2とし、これらは拡張カルマンフィルタで統合される。車輪型ロボットで人工環境を走行することを想定しており、各姿勢角度変化がある程度急激なものなく、カルマンフィルタで十分扱えると判断する。

**2・1 主法線抽出** センサで得られる点群に対して主成分分析を用いて、各注目点に対して法線ベクトルを算出する。この法線群から、信頼性の高い鉛直面を持つ法線を以下の条件で抽出する。

- 注目点が持つ近傍点の数が十分多い（密度が高い）。
- 法線と1ステップ前の推定重力ベクトルとの角度が直角に十分近い。1ステップ前の推定重力ベクトルをもとに鉛直面であるかを判定している。
- 法線に垂直な平面と近傍点群の二乗誤差が十分小さい（平面度が高い）。

抽出された法線群でガウス球<sup>(2)</sup>を生成し、球内の点群に対して階層クラスター分析を適用する。なお本手法では、適当な法線を反転させガウス球をさらに半球

<sup>\*1</sup> 明治大学理工学部機械工学科（〒214-8571 神奈川県川崎市多摩区東三田1-1-1）ee53031@meiji.ac.jp

<sup>\*2</sup> 明治大学理工学部機械工学科（〒214-8571 神奈川県川崎市多摩区東三田1-1-1）ykuroda@isc.meiji.ac.jp

のみに反映している。つまり、平行で向かい合う平面は同じ法線ベクトルを持つようとする。クラスタリング終了後に、一定回数以上統合されなかったガウス球の点または点群は外れ値として除去する。

**2.2 重力方向推定** クラスタリングされた法線(ガウス球内の点)を用いてロボット座標系における単位重力ベクトル  $\mathbf{G}$  を推定する。クラスタリングされた法線の数に応じて以下のように算出する。ただし、算出後に、1ステップ前の重力方向をもとに、算出したベクトルの向きを適当に決める必要がある。

- 法線の数 : 3 以上

法線をガウス球における点として扱い、この点群に対する主成分分析で得た法線を重力ベクトルとする。ただし、クラスタが持つ点群の数で重み付けを行っている。

- 法線の数 : 2

2本の法線ベクトルの外積を重力ベクトルとする。

- 法線の数 : 1

1ステップ前の推定重力ベクトル  $\mathbf{G}'$  を用いて、式(1)のように部分的に推定重力ベクトルを補正し算出する。観測された法線ベクトルを  $\mathbf{N}$  とする。

$$\mathbf{G} = \frac{\mathbf{G}' - (\mathbf{G}' \cdot \mathbf{N})\mathbf{N}}{\|\mathbf{G}' - (\mathbf{G}' \cdot \mathbf{N})\mathbf{N}\|} \quad (1)$$

- 法線の数 : 0

重力方向を推定することが出来ない。

### 3. 実験

図1に明治大学生田キャンパス内の実験環境の航空写真を示す。一周約250m、高低差約3mのコースを、スタートの位置、姿勢と、ゴールの位置、姿勢が同じになるようにロボットをコントローラー操作し3周させる。提案手法および比較手法で姿勢を推定し、並進運動は各手法共通でホイールオドメトリのみを用いる。姿勢推定の比較手法としてジャイロオドメトリと、LSD-SLAM<sup>(3)</sup>を行う。LSD-SLAMでも姿勢のみを推定し共通のホイールオドメトリと統合する。また、本実験ではLSD-SLAMでのループクローズは行っていない。提案手法の観測の一部となるSLAMにも、LSD-SLAMを用いる。ロボットに搭載され使用するセンサは、Velodyne HDL-32E、Xsens MTi10、RealSense D435、ホイールエンコーダである。RealSenseはRGB-Dカメラであるが、LSD-SLAMのために画像のみを利用しているため、深度情報は利用していない。

表1に、3周後の推定位置姿勢と初期位置姿勢との誤差を示す。ユークリッド距離で表した位置推定誤差に関して、比較手法に比べ提案手法が小さいことから、走行中の姿勢推定の誤差も提案手法の方が小さいと言

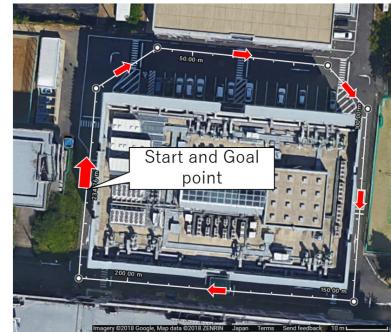


Fig. 1 The experimental environment

Table 1 Return position and pose errors

error in...	Proposed method	LSD-SLAM	Gyrodometry
$x[m]$	-1.877	-1.517	-10.296
$y[m]$	1.180	1.802	-1.430
$z[m]$	-0.446	4.216	18.534
$Euc.distance[m]$	2.261	4.829	21.250
$roll[deg]$	-0.045	5.171	0.261
$pitch[deg]$	-0.897	-18.218	-4.344
$yaw[deg]$	1.358	-2.735	15.431

える。特に、Z軸(鉛直)方向の並進誤差が小さく抑えられており、壁面を用いたロール、ピッチ補正による結果だと考察する。

### 4. 結 言

従来の慣性センサとSLAMによる移動ロボットの姿勢推定に加え、鉛直に建てられている建造物の壁に対する相対姿勢を利用する姿勢推定法を提案した。実験結果より、本提案手法の、蓄積誤差に対する補正の有用性が示された。

### 謝 辞

本研究の一部は、NEDO次世代人工知能・ロボット中核技術開発事業による支援を受けた。また、実験に使用した移動ロボットはSEQSENSE株式会社より提供を受けたものである。ここに篤く御礼申し上げる。

### 参 考 文 献

- (1) L. von Stumberg, V. Usenko and D. Cremers, “Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization”, in *International Conference on Robotics and Automation*, (2018).
- (2) B.K.P. Horn, “Extended gaussian images”, *Proceedings of the 1984 IEEE*, (1984), pp.1671–1686.
- (3) J. Engel, J. Stueckler and D. Cremers, “LSD-SLAM: Large-Scale Direct Monocular SLAM”, *Computer Vision-ECCV 2014, European Conference on Computer Vision*, (2014), pp.834–849.

## 6-DoF EKF SLAM with global planar features in artificial environments

Ryota Ozaki<sup>1</sup> and Yoji Kuroda<sup>2</sup>

**Abstract**—This paper presents online SLAM for self-localization of mobile robots in artificial environments. This proposed method exploits global planar features as landmarks in extended Kalman filter (EKF). The main purpose of using global planar features is reducing accumulative error of the estimation. Planes are extracted from point cloud of 3-D LiDAR as normals. These normals are projected onto “depth-Gaussian sphere”. Those points from each plane are concentrated in one place on the sphere since planes has many normals which are almost same directions. These concentrations of points are deemed as planar features. The state vector of EKF has states of both a robot and landmarks. Prediction steps compute integration of angular velocity from gyroscope and linear velocity from wheel encoders, respectively. Observation steps update the states by associating observed planes and landmark planes. Area of landmark planes are also expanded with every association. Observed planes which are not associated with any landmarks become new landmarks, and the state vector is augmented. Similar landmarks are merged as necessary. To avoid mismatching between observations and landmarks, candidates for the association are selected by some conditions. To evaluate the proposed method, an experiment with an actual robot was performed. It shows the method suppresses accumulative error of self-localization by using the landmarks.

### I. INTRODUCTION

Estimating the pose of a robot in the surrounding environment is one of the classic problems of mobile robotics. In a known environment, the robot can self-localize by matching sensor information at the moment to the prior information of the environment. Especially, many methods using maps as prior information have been proposed [1]. In those methods, accuracy of the map is important. However, mapping requires accurate estimation of the robot’s pose. This leads a dilemma: for self-localization, the robot requires a map, but to build such a map, the pose of the robot must be known [2]. A solution of this is SLAM (Simultaneous Localization And Mapping) [3]. Many kinds of SLAM have been developed. This paper focuses on online (real-time) SLAM because localizing the pose and correcting estimation online is required when the robot runs in unknown environments. SLAM with scan matching method such like ICP scan matching [4], NDT scan matching [5], [6] is one of well-known online SLAM. Zhang [7] has proposed a matching method based on edge and planar features with good results. Using these features achieves low-drift and low-computational complexity. On the other hand, it is difficult for scan matching methods to correct accumulative error because it integrates relative

pose transformation to the initial pose. On another hand, landmark-based SLAM can suppress error while the robot is observing the landmarks. Taguchi [8] has proposed a method which associates observation and landmarks, and applies SVD to estimate the pose of the robot. But using SVD means noise of observation is not considered. Landmark-based SLAM implemented with EKF is well known [9]. Exploiting planes as landmarks is a well known method in the area of visual SLAM [10], [11], [12]. However they do not describe how to handle landmarks which the robot pass by and can not be observed. It means it is not considered how to deal with the situation that robot comes back to known place, and how to avoid false matching. Some methods use planar landmarks with Manhattan world assumption [13], [14]. The assumption assumes that planes in the environment are orthogonal to each other. This assumption has low versatility although it linearizes the system and makes estimation easier.

To address these issues above, this paper proposes EKF SLAM with global planar features as landmarks. And data association including the situation that the robot comes back to known landmarks is implemented. Note that the proposed method does not use any prepared maps nor models of the environment. Information of the planar landmarks such like position and orientation are also unknown. And Manhattan world assumption is not used in this method, which means any planes can be used in this method.

Main contributions of this paper are summarized here:

- A method of extracting planes from point cloud as normals is described.
- EKF framework with planar landmarks is described.
- Data association between sensor observations and landmarks including the revisit situation is described.
- Condition setting for avoiding false matching is described.

### II. 6-DOF EKF SLAM WITH GLOBAL PLANAR FEATURES IN ARTIFICIAL ENVIRONMENTS

The system configuration diagram of the proposed method is shown in Fig. 1. The method is based on landmark-based SLAM [3]. Prediction step and update step are repeated in EKF. Date association between observations and registered landmarks is needed for the update process. Landmarks means planar features in this method. The planar features are extracted from the point cloud.

#### A. Extraction of planes from point cloud

1) *Generation of normal-cloud*: Normal-cloud  $\mathbf{N}$  is generated by applying principal component analysis (PCA)[15] to the local neighbor points of each query point in point

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

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

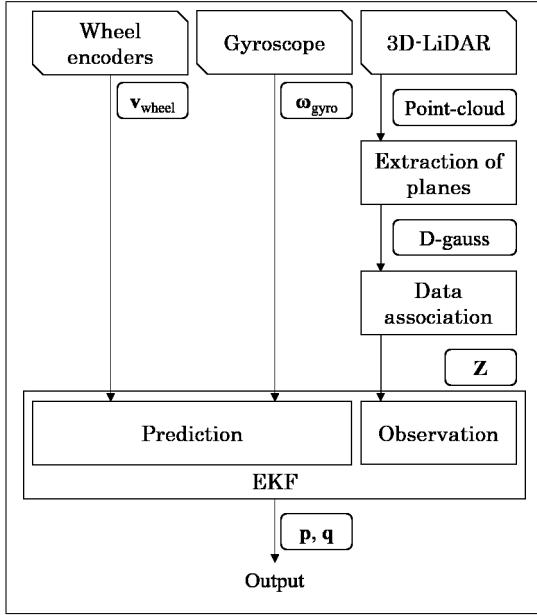


Fig. 1: System architecture

cloud that is obtained with 3-D LiDAR. The neighbor points (query point cloud)  $\mathbf{C}_{\text{query}}$  are searched by kd-tree[16] with searching radius. This searching radius is set larger at farther point since point density is more sparse in farther area.

$$\mathbf{C}_{\text{query}} = [\mathbf{c}_0 \ \cdots \ \mathbf{c}_{n_{\mathbf{C}_{\text{query}}}}], \quad \mathbf{c}_i = (c_{i,x} \ c_{i,y} \ c_{i,z}) \quad (1)$$

$$\mathbf{N} = [\mathbf{n}_0 \ \cdots \ \mathbf{n}_{n_{\mathbf{N}}}], \quad \mathbf{n}_i = (n_{i,a} \ n_{i,b} \ n_{i,c} \ n_{i,d}) \quad (2)$$

where  $\mathbf{c}_i$  denotes a point, and  $\mathbf{n}_i$  denotes parameters of the plane (i.e.  $n_{i,a}x + n_{i,b}y + n_{i,c}z + n_{i,d} = 0$ ).

2) *Selection from normal-cloud:* Query point clouds which have high flatness are picked up from the normal-cloud with their third eigenvector (normal) by the conditions below.

- The number of query points  $n_{\mathbf{C}_{\text{query}}}$  is large enough.

$$n_{\mathbf{C}_{\text{query}}} > \text{TH}_{n_{\mathbf{C}_{\text{query}}}} \quad (3)$$

- Fitting error between the plane and neighbor points  $e$  is small enough.

$$e = \sum_{i=0}^n \frac{|n_a c_{i,x} + n_b c_{i,y} + n_c c_{i,z} + n_d|}{\|\mathbf{n}\|} < \text{TH}_e \quad (4)$$

3) *Generation of depth-Gaussian sphere:* All initial points of selected normals  $\hat{\mathbf{N}} (\in \mathbf{N})$  are moved to one origin and generate point cloud  $\mathbf{C}_{\text{d-gauss}}$ . Shimizu [17] calls this point cloud “depth-Gaussian sphere” (cf. Gaussian sphere [18]). Fig. 2 shows an outline drawing of depth-Gaussian sphere.

$$\mathbf{C}_{\text{d-gauss}} = [\mathbf{c}_{\text{d-gauss}0} \ \cdots \ \mathbf{c}_{\text{d-gauss}_{n_{\hat{\mathbf{N}}}}}] \quad (5)$$

$$\mathbf{c}_{\text{d-gauss}}_i = -n_{i,d} (n_{i,a} \ n_{i,b} \ n_{i,c})$$

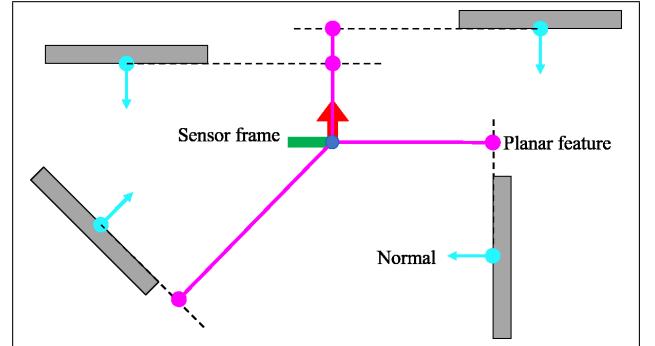


Fig. 2: Depth-Gaussian sphere

4) *Clustering in depth-Gaussian sphere:* Euclidean clustering is applied to the point cloud on depth-Gaussian sphere. And clusters which has enough many numbers of members are extracted as plane. Centroid of each extracted cluster is used as planar feature below.

### B. EKF framework

The state of both the robot and planar landmarks are estimated simultaneously in this EKF. The state vector  $\mathbf{x}$  consists of the state of them as Eq. (6). Prediction process is done with input of wheel encoders and gyroscope. Observation process is done with observation of planar landmarks.

$$\begin{aligned} \mathbf{x} &= (\mathbf{p}^T \ \mathbf{q}^T \ \mathbf{m}_0^T \ \cdots \ \mathbf{m}_n^T)^T \\ \mathbf{p} &= (x_r \ y_r \ z_r)^T, \quad \mathbf{q} = (\phi_r \ \theta_r \ \psi_r)^T \\ \mathbf{m}_i &= (x_{\text{lm},i} \ y_{\text{lm},i} \ z_{\text{lm},i})^T \end{aligned} \quad (6)$$

where  $\mathbf{p}$  denotes the position of the robot,  $\mathbf{q}$  denotes the posture of the robot, and  $\mathbf{m}_i$  denotes the position of the landmark.

1) *Prediction with wheel encoder and gyroscope:* Linear velocity measured with wheel encoder  $\mathbf{v}_{\text{wheel}}$  and angular velocity measured with gyroscope  $\omega_{\text{gyro}}$  are transformed from the local coordinate to the global coordinate.

$$\mathbf{u}_{\text{wheel}} = \mathbf{v}_{\text{wheel}} = (v_x \ 0 \ 0)^T \quad (7)$$

$$\mathbf{u}_{\text{gyro}} = \omega_{\text{gyro}} = (\omega_x \ \omega_y \ \omega_z)^T \quad (8)$$

$$f\{\mathbf{x}_k\} = \mathbf{x}_k + \begin{pmatrix} \mathbf{Rot}_{xyz}^{l \rightarrow g}\{\mathbf{q}_k\} \mathbf{u}_{\text{wheel},k} \Delta t \\ \mathbf{Rot}_{\phi\theta\psi}^{l \rightarrow g}\{\mathbf{q}_k\} \mathbf{u}_{\text{gyro},k} \Delta t \\ \mathbf{0}_{3 \times n} \end{pmatrix} \quad (9)$$

where  $\mathbf{Rot}_{xyz}^{l \rightarrow g}$  is the rotation matrix which transforms the point from the local frame to the global one, and  $\mathbf{Rot}_{\phi\theta\psi}^{l \rightarrow g}$  is the rotation matrix which transforms the angles from the local frame to the global one.

2) *Observation of planar landmarks:* Observation process is done when observations of planar features are associated with known landmarks. When the observations is not associated with any landmarks, the state vector is augmented and the state of the new landmark is added. How to associate

between observations and landmarks is described in next section.

$$\mathbf{Z} = (\mathbf{z}_0^T \quad \dots \quad \mathbf{z}_n^T)^T \quad (10)$$

$$\mathbf{z}_i = -n_{i,d} (n_{i,a} \quad n_{i,b} \quad n_{i,c})^T$$

$$\mathbf{Z}_p = (\mathbf{z}_{p_0}^T \quad \dots \quad \mathbf{z}_{p_n}^T)^T \quad (11)$$

$$\mathbf{z}_{p_i} = h_i\{\mathbf{x}_k\} = \text{Rot}_{xyz}^{g \rightarrow l}\{\mathbf{q}_k\}(\mathbf{m}_{i,k} - \frac{\mathbf{p}_k \cdot \mathbf{m}_{i,k}}{\|\mathbf{m}_{i,k}\|^2} \mathbf{m}_{i,k})$$

where  $\mathbf{Z}$  is the observation vector, and  $\mathbf{Z}_p$  is the predicted observation vector. Fig. 3 shows the function of  $h_i\{\mathbf{x}\}$ .

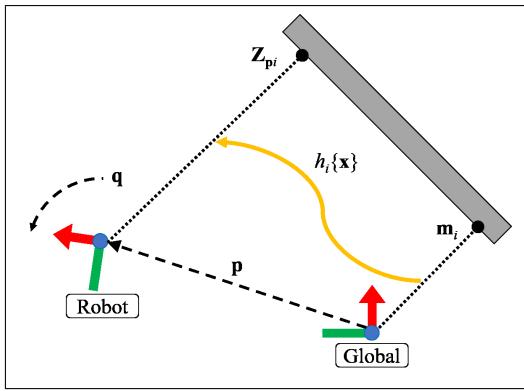


Fig. 3: Predicted observation

### C. Data association

1) *Search for a corresponding landmark:* Each registered landmark searches observation which has minimum Mahalanobis distance  $d_{\text{Mahalanobis,min}}$ . When the distance is smaller than the threshold, the observation is associated with the known landmark. When it is larger, the observation is registered as a new landmark.

$$d_{\text{Mahalanobis,min}} = \mathbf{y}_i^T \mathbf{S}_i^{-1} \mathbf{y}_i < \text{TH}_{d_{\text{Mahalanobis}}} \quad (12)$$

$$\mathbf{y}_i = \mathbf{z}_i - \mathbf{z}_{p_i}, \quad \mathbf{S} = \mathbf{J}_h \mathbf{P} \mathbf{J}_h^T + \mathbf{R}$$

where  $\mathbf{J}_h$  is the Jacobian of the vector  $\mathbf{z}_p$ ,  $\mathbf{P}$  is the covariance matrix, and  $\mathbf{R}$  is the noise matrix of observation.

2) *Registration of a new landmark:* A new landmark is registered and initialized with the information below:

- Own coordinate (origin)  
The point which the landmark is observed at the first time becomes its origin of the own coordinate.
- Observed range  
Each landmark records the range which the robot has observed it before in the coordinate of the landmark.
- Direction of normal  
Planes can have two direction of normal. But only one side of the planes can be observed in the real world because walls and other things have thickness. Therefore the direction of the plane is registered when the landmark is initialized.

3) *Update of information of associated landmark:* When observation is associated with the observation, the information of the landmark below is updated.

- Orientation of coordinate
- Observed range

4) *Merge of landmarks:* Landmarks are merged when two or more landmarks find a same observation as a corresponding one. The oldest landmark which is observed earliest is remained, and the others are absorbed to the oldest one. The range which the robot has observed the landmark is also merged. This merge can happen when the robot comes back to the known place.

However, landmarks are not merged when these landmarks have been observed at the same time at prior step. At that time, only the landmark whose Mahalanobis distance  $d_{\text{Mahalanobis,min}}$  is smaller is associated, and the others have no pair.

5) *Narrowing down the candidates:* In order to avoid false association, registered landmarks are narrowed down every step before association process above. Geometric constraints that the method exploits are here:

- Only one side of the plane is visible and it is not visible from the other side.
- The landmark is not visible from a far position from the last position of the robot where the landmark is observed (observed range) like Fig. 4.

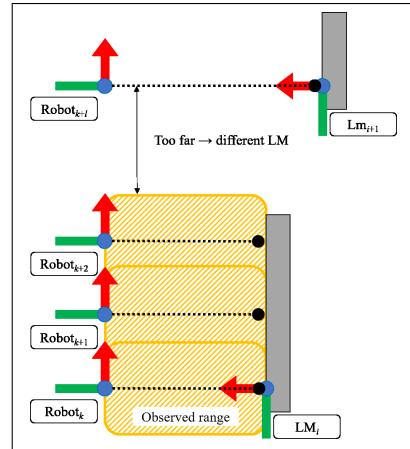


Fig. 4: Observed range

## III. EXPERIMENTS

### A. Experimental outline

In this experiment, the 6-DoF pose (position and attitude) of the robot was estimated by the proposed method and comparative approaches while the robot moved. It is hard to get all ground truth of the pose while the robot is moving. Hence, the robot was driven back to the exact starting position. Thus, the return position error and the return attitude error were evaluated instead of the whole trajectory.

## B. Experimental conditions

1) *Hardware*: The experimental mobile robot is shown in Fig. 5. It has Velodyne HDL-32E, Xsens MTi30 and wheel encoders.

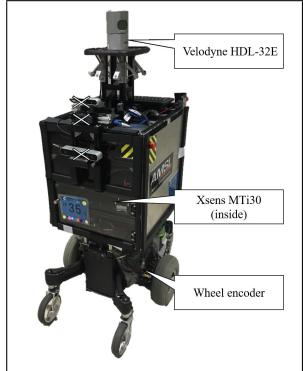


Fig. 5: The experimental robot

2) *Environment*: The experimental floor map is shown in Fig. 6. An planar obstacle was put in the environment in order to make sure that the proposed method does not use Manhattan world assumption. The robot was driven for three rounds of this course.

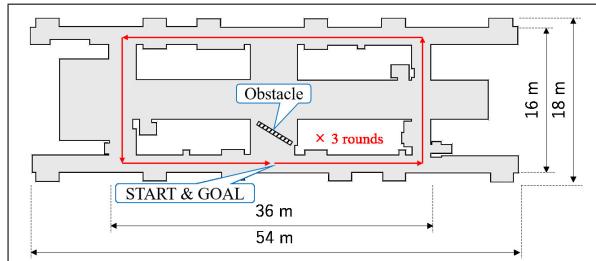


Fig. 6: The experimental floor map

### 3) Comparative approaches:

- Gyrodometry [19]
- LOAM [7]

## C. Experimental result

Trajectories estimated by the proposed method and the comparative approaches on x-y plane are shown in Fig. 7a. The trajectories on x-z plane are shown in Fig. 7b. The return position error and the return attitude error are shown in Table I. The proposed method estimated the pose of the robot with less error in Euclidean distance compared with the others. It was stable by using global planar features. The method could correct the estimation as long as the robot observes the registered planar landmarks. And some landmarks were merged when the robot came back to the start point from the other side. On the other hand, it was hard for gyrodometry and LOAM to correct accumulative error while it was driven. Although LOAM estimated transformation of the pose well each step, once it detected bad matching, it was hard to correct, which is comes from the feature of scan matching method. The average time consumption of the proposed

method is 0.113 sec for extraction the planar feature, and 0.009 sec for the association and updating the state vector.

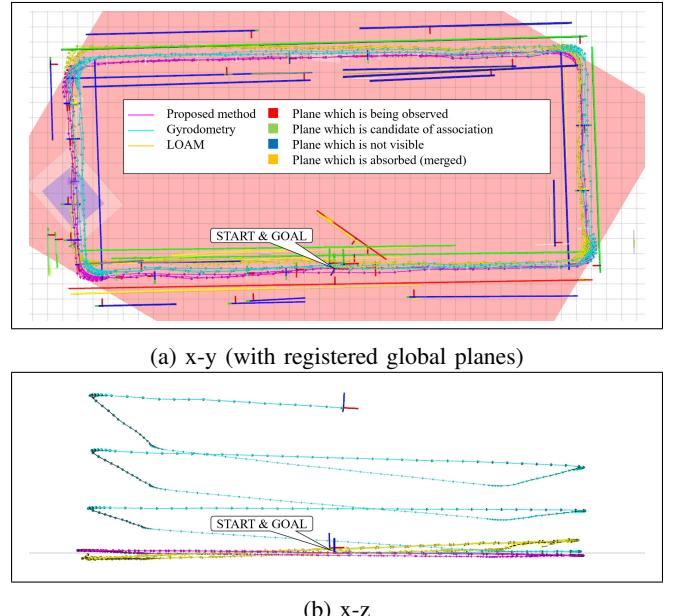


Fig. 7: The estimated trajectories

TABLE I: Return position error and the return attitude error

error in...	Proposed method	Gyrodometry	LOAM
x[m]	-0.042	+0.600	-0.338
y[m]	+0.019	+0.314	+0.359
z[m]	+0.004	+9.753	+0.341
$d_{\text{Euc}}[\text{m}]$	0.046	9.776	0.600
$\phi[\text{deg}]$	+0.03	+2.39	-0.87
$\theta[\text{deg}]$	+0.08	+2.32	-2.05
$\psi[\text{deg}]$	+1.35	+2.17	+0.49

## IV. CONCLUSIONS AND FUTURE WORK

6-DoF EKF SLAM with global planar features in artificial environments was proposed. By measuring the position of planar landmarks, 6-Dof robot pose and the position of associated global planes are updated. The experiment showed that the proposed method had less accumulative error of estimation than comparative approaches.

Since bundle adjustment is not implemented although the proposed method merges landmarks when the revisits known places, the future work of this paper is developing a method to fix motion estimation drift by closing the loop. Another future work is applying the proposed method to outdoor environments. Outdoor ground is not complete plane. Therefore the method often does false matching so far.

## 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] 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, pp.312–328, 2007.
- [2] P. Skrzypczynski, Simultaneous localization and mapping: A feature-based probabilistic approach, *International Journal of Applied Mathematics and Computer Science*, Vol.19, No.4, pp.575–588, 2009.
- [3] S. Thrun, W. Burgard and D. Fox, *probabilistic robotics*, The MIT Press, pp.309–336, 2005.
- [4] S. Rusinkiewicz and M. Levoy, Efficient Variants of the ICP Algorithm, *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pp.145–152, 2001.
- [5] P. Biber and W. Straßer, The Normal Distributions Transform: A New Approach to Laser Scan Matching, *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol.3, pp.2743–2748, 2003.
- [6] M. Magnusson, A. Lilienthal and T. Duckett, Scan registration for autonomous mining vehicles using 3D NDT, *Journal of Field Robotics*, Vol.24, No.10, pp.803–827, 2007.
- [7] J. Zhang and S. Singh, LOAM: Lidar Odometry and Mapping in Real-time, in *Robotics: Science and Systems Conference*, pp.161–195, 2014.
- [8] Y. Taguchi, Y.D. Jian, S. Ramalingam and C. Feng, Point-Plane SLAM for Hand-Held 3D Sensors, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pp.5182–5189, 2013.
- [9] S. Huang and G. Dissanayake, Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM, *IEEE Transactions on Robotics*, Vol.23, No.5, pp.1036–1049, 2007.
- [10] J. Martinez-Carranza and A. Calway, Unifying Planar and Point Mapping in Monocular SLAM, *Proceedings of the British Machine Vision Conference (BMVC)*, pp.43.1–43.11, 2010.
- [11] A. P. Gee, D. Chekhlov, WW. Mayol-Cuevas, A. Calway, Discovering Planes and Collapsing the State Space in Visual SLAM, *Proceedings of British Machine Vision Conference*, pp.1–10, 2007.
- [12] A. P. Gee, D. Chekhlov, A. Calway, W. Mayol-Cuevas Discovering Higher Level Structure in Visual SLAM, *IEEE Transactions on Robotics*, Vol.24, No.5, pp.980–990, 2008.
- [13] P. Kim, B. Coltin and H. J. Kim, Linear RGB-D SLAM for Planar Environments, *Proceedings of The European Conference on Computer Vision (ECCV)*, pp.333–348, 2018.
- [14] K. Takeuchi, M. Hashimoto and K. Takahashi, Laser-Based-3D Mapping by Human Walking in an Indoor Environment, *The Science And Engineering Review Of Doshisha University*, Vol.53, No.2, pp.84–91, 2012 (in Japanese).
- [15] M. Pauly, M. Gross and L.P. Kobbelt, Efficient simplification of point-sampled surfaces, *Proceedings of the conference on Visualization*, pp.163–170, 2002.
- [16] J.L. Bentley, Multidimensional binary search trees used for associative searching, *Communications of the ACM*, Vol.18, No.9, pp.509–517, 1975.
- [17] S. Shimizu and Y. Kuroda, High-speed registration of point clouds by using dominant planes, *Proceedings of the 19th Robotics Symposia*, pp.453–458, 2014 (in Japanese).
- [18] B.K.P. Horn, Extended gaussian images, *Proceding of the IEEE*, Vol.72, No.12, pp.1671–1686, 1984.
- [19] J. Borenstein and L. Feng, Gyrodometry: a new method for combining data from gyros and odometry in mobile robots, *Proceedings of IEEE International Conference on Robotics and Automation*, pp.423–428, 1996.

# 人工環境における平面特徴量を用いたランドマーク SLAM

尾崎 亮太 <sup>\*1</sup>, 黒田 洋司 <sup>\*2</sup>

## Landmark SLAM using planar features in artificial environments

Ryota OZAKI<sup>\*1</sup> and Yoji KURODA<sup>\*2</sup>

<sup>\*1\*2</sup> Mechanical Engineering Program, Graduate School of Science and Technology, Meiji University  
1-1-1 Higashimita, Kawasaki Tama-ku, Kanagawa 214-8571, Japan

This paper presents online SLAM for self-localization of mobile robots in artificial environments. This proposed method exploits global planar features as landmarks in extended Kalman filter (EKF). Planar features are extracted from point-cloud of 3D LiDAR by clustering normals which are computed by principal component analysis (PCA). These observed features are associated with registered landmarks. Some geometric constraints are used in this association in order to avoid false matching. Observed features which are not associated with any landmarks become new landmarks. The state vector of EKF has states of both a robot and landmarks. Prediction steps compute integration of angular velocity from gyroscope and linear velocity from wheel encoders, respectively. Observation steps update states by using association between observed features and landmarks. The state vector is augmented when the new landmarks are registered. Similar landmarks are merged as necessary. To evaluate the proposed method, an experiment with an actual robot was performed. It showed the method suppresses accumulative error of localization by using the landmarks.

**Key Words :** Mobile robot, Artificial environment, Self-localization, Landmark-SLAM, EKF-SLAM

### 1. 緒 言

移動ロボットの分野では、未知の環境を走行するとき、自己位置推定と地図生成を同時に使う SLAM<sup>(1)</sup>がよく用いられる。ICP<sup>(2)</sup>や NDT<sup>(3)</sup>を用いたスキャンマッチングは、典型的な SLAM の手法の 1 つだ。Zhang<sup>(4)</sup>は、低ドリフトと低計算量を目的として、エッジと平面を特徴量として用いるスキャンマッチングを提案した。しかしながら、これらのようなスキャンマッチングは、相対変化量を積算するので、蓄積誤差を補正するのが難しい。一方、ランドマーク SLAM<sup>(1)</sup>は、ランドマークを観測している限り、蓄積誤差を補正しやすい手法である。特に、EKF を用いたランドマーク SLAM はよく知られている<sup>(5)</sup>。また、カメラを用いた Visual SLAM の分野では、平面特徴量を用いた手法が提案されている<sup>(6)(7)</sup>。しかし、これらの論文では、ロボットが通り過ぎて観測できなくなったランドマークの扱いに関して明記されていない。

上記の問題を解決するため、本論文では、人工環境の平面をランドマークとして用い、通り過ぎて観測で

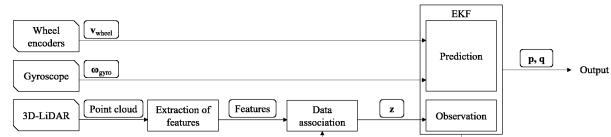


Fig. 1 System architecture.

きなくなったランドマークの情報を残すことができる EKF-SLAM を提案する。

### 2. 平面をランドマークとして用いる SLAM

図 1 に提案手法のシステム図を示す。

**2.1 平面特徴量の抽出** まず、3D-LiDAR の点群の各局所点群に主成分分析 (PCA) を適用することで、法線群を生成する。その法線群から、条件を設定することで、高い平面度を持つ平面のみを選定する。選定された法線群に対して条件付きユークリッドクラスタ分析を適用し、メンバの数が十分多いクラスタを選定する。各クラスタのそれぞれの法線の始点を、原点に移動させて、深さを付加して法線の各成分をユークリッド空間に写像する（深さ付きガウス球）。Fig.2 に、深さ付きガウス球の概要図を示す。深さ付きガウス球に写像された各クラスタの重心を特徴量として抽

<sup>\*1</sup> 明治大学理工学研究科機械工学専攻 (〒 214-8571 神奈川県川崎市多摩区東三田 1-1-1) cc192021@meiji.ac.jp

<sup>\*2</sup> 明治大学理工学研究科機械工学専攻 (〒 214-8571 神奈川県川崎市多摩区東三田 1-1-1) ykuroda@meiji.ac.jp

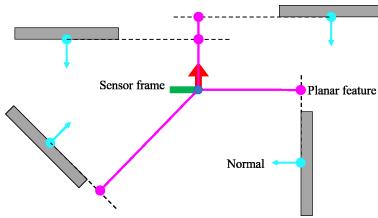


Fig. 2 Depth-Gaussian sphere

出する。

**2.2 EKF の構成** 状態ベクトルはロボットとランドマークの状態で構成され、EKF でそれを推定する。ジャイロスコープとホイールエンコーダを用いて予測を行う。また、観測された特徴量とランドマークの対応付けで更新を行う。対応付けされなかった特徴量は新たなランドマークとして登録される。

**2.3 データアソシエーション** 観測された特徴量とランドマークとのマハラノビス距離に閾値を設定することで、それらの対応付けを行う。ただし、幾何学的拘束条件を用いることで誤マッチングへの対策を行う。具体的には、1) 平面の裏側は観測できない拘束と、2) 距離が遠すぎるランドマークは観測できない拘束である。これを適用することで、各ステップで観測できそうなランドマークのみを一時的に抽出してから計算することができる。つまり、走行とともに登録したランドマークの数が増えても、各ステップでのEKF の計算コストはほとんど変わらない。

### 3. 実験

移動ロボットを走行させて、ロボットの位置と姿勢を推定する実験を行った。その推定は、提案手法と従来手法を用いてそれぞれ行われた。ただし、走行コースの全体でロボットの位置と姿勢の真値を計測することは難しいため、走行終了時（ゴール地点）の推定値のみを評価した。つまり、コース全体で発生した蓄積誤差が、走行終了時の誤差として表れると仮定して評価した。さらに、スタート地点とゴール地点を同一にし、ロボットがスタート地点に戻ってくるように操作することで、簡易的に走行終了時の真値を  $(x, y, z, \phi, \theta, \psi) = \mathbf{0}$  とした。

ロボットに搭載したセンサは、3D-LiDAR (Velodyne HDL-32E)，IMU (Xsens MTi30)，ホイールエンコーダである。1周 100 m のコースをロボットに 3 周走行させ、それを 5 セット繰り返した。従来手法として、ジャイロオドメトリと LOAM<sup>(4)</sup>を用いた。

表 1 に 5 セットの実験における誤差の絶対値の平均値を示す。比較手法に比べ、提案手法の誤差がより小さい結果となった。

Table 1 Average in 5 sets of the driving.

error in...	Proposed method	Gyroodometry	LOAM
$x[m]$	0.058	1.165	0.192
$y[m]$	0.024	0.370	0.094
$z[m]$	0.054	6.373	0.200
$d_{\text{Euc}}[m]$	0.094	6.504	0.314
$\phi[\text{deg}]$	0.24	1.10	1.30
$\theta[\text{deg}]$	0.11	1.73	2.16
$\psi[\text{deg}]$	0.95	7.12	1.34

### 4. 結言

人工環境の平面をランドマークとして用いる EKF-SLAM を提案した。この手法では、ランドマークを観測することで、ロボットの位置、姿勢とランドマークの位置が更新される。観測された平面特徴量とランドマークの対応付けでは、誤マッチングを防ぐよういくつかの条件を設定した。評価実験では、提案手法の誤差は従来手法の誤差より小さい結果となった。

### 謝辞

本研究の一部は、国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の、次世代人工知能・ロボット中核技術開発事業による支援を受けた。ここに篤く御礼申し上げる。

### 参考文献

- S. Thrun, W. Burgard and D. Fox: “probabilistic robotics”, *The MIT Press*, pp.309–336(2005).
- S. Rusinkiewicz and M. Levoy: “Efficient Variants of the ICP Algorithm”, *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pp.145–152(2001).
- P. Biber and W. Straßer: “The Normal Distributions Transform: A New Approach to Laser Scan Matching”, *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol.3, pp.2743–2748(2003).
- J. Zhang and S. Singh: “LOAM: Lidar Odometry and Mapping in Real-time”, *Proceedings of Robotics: Science and Systems Conference*, pp.161–195(2014).
- S. Huang and G. Dissanayake: “Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM”, *IEEE Transactions on Robotics*, Vol.23, No.5, pp.1036–1049(2007).
- J. Martinez-Carranza and A. Calway: “Unifying Planar and Point Mapping in Monocular SLAM”, *Proceedings of the British Machine Vision Conference (BMVC)*, pp.43.1–43.11(2010).
- A. P. Gee, D. Chekhlov, WW. Mayol-Cuevas, A. Calway: “Discovering Planes and Collapsing the State Space in Visual SLAM”, *Proceedings of British Machine Vision Conference*, pp.1–10(2007).

# DNN-based self-attitude estimation by learning landscape information

Ryota Ozaki<sup>1</sup> and Yoji Kuroda<sup>2</sup>

**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-

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

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

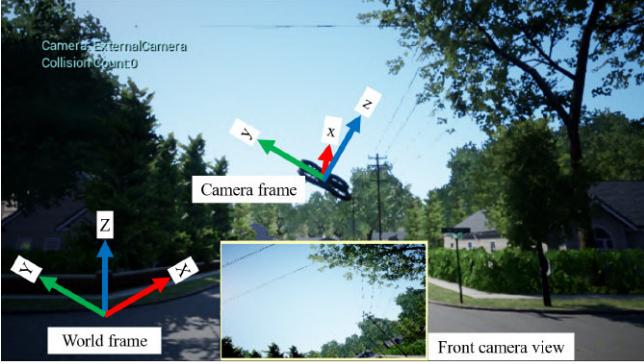


Fig. 1: Screenshot of AirSim with coordinate description. An IMU and a camera are equipped to the drone in the simulator. A purpose of this work is estimating a gravity vector in the camera frame from a front camera image.

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  $\mathbf{g}$  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  $\phi$  and pitch  $\theta$  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  $\alpha$  is limited as [-10 deg, 10 deg]. The image is resized to  $224 \times 224$ . RGB values are normalized. In this work, this normalization is done following  $\text{mean} = (0.5, 0.5, 0.5)$  and  $\text{std} = (0.5, 0.5, 0.5)$ . Fig.3 shows an example of data augmentation.

2) *Gravity vector (label)*: A gravity vector in the camera frame is rotated according to  $\alpha$ . 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.

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

### 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

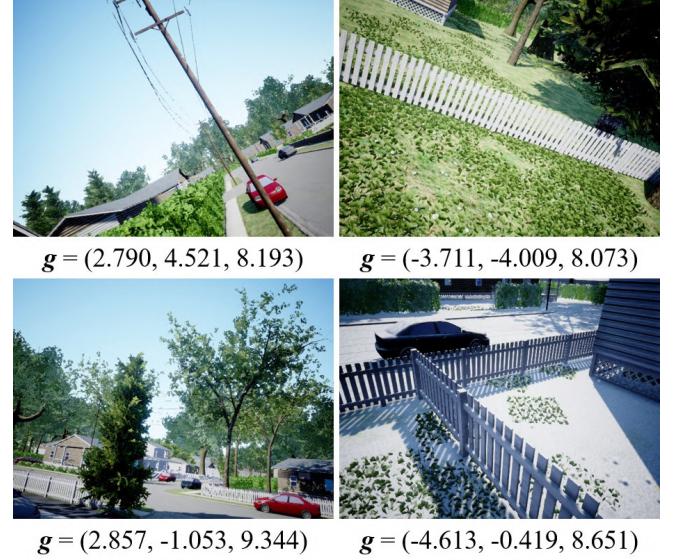


Fig. 2: Examples of datasets. The dataset consists of images and correspond gravity vectors  $\mathbf{g}[\text{m}/\text{s}^2]$  in the camera frame. These data are collected in ‘‘Neighborhood’’ of AirSim. The camera pose and weather parameters are randomized for creating a dataset. Road in the lower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



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

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

$$\boldsymbol{\mu} = \frac{(\mu_{gx}, \mu_{gy}, \mu_{gz})^\top}{|(\mu_{gx}, \mu_{gy}, \mu_{gz})^\top|} \quad (2)$$

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

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.

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

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  $P(\mathbf{x}_{\text{label}}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ , where  $\mathbf{x}_{\text{label}}$  ( $= \mathbf{g}_{\text{trans}}$ ) is a label of the dataset. The total probability density  $P_{\text{total}}$  of a dataset  $D_{\text{label}} = [\mathbf{x}_{\text{label}_0}, \dots, \mathbf{x}_{\text{label}_i}, \dots, \mathbf{x}_{\text{label}_n}]$  is computed by multiplying as Eq.5.

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

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

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

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

$$f(\mathbf{w}) = -P_{\log_{\text{total}}} \quad (7)$$

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

### F. Optimization

Adaptive Moment Estimation (Adam)[20] is used to optimize the parameters. The learning rates are set as  $lr_{\text{CNN}} = 0.00001$ ,  $lr_{\text{FC}} = 0.0001$ , where  $lr_{\text{CNN}}$  is a value for CNN layers,  $lr_{\text{FC}}$  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 <sup>2</sup> ]	-6.4991	-5.7436
Regression w/ L2 normalization [m <sup>2</sup> /s <sup>4</sup> ]	0.0011	0.0031
Regression w/o L2 normalization [m <sup>2</sup> /s <sup>4</sup> ]	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  $\beta$  in Eq.8 expresses uncertainty of the prediction, samples with small variance are selected with a threshold  $TH_{\beta}$ . In this validation, the threshold is set as  $TH_{\beta} = \frac{1}{n} \sum_{i=0}^n \beta_i$ , where  $n$  is a number of samples in the testing dataset.

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

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<sup>2</sup>, in order to minimize the loss. This information is not required to estimate attitude  $\phi$ ,  $\theta$ .

### 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  $\phi$  and pitch  $\theta$  of the camera pose in the gravitational coordinate are estimated by using  $\boldsymbol{\mu}$ .

$$\phi = \tan^{-1} \frac{\mu_{gy}}{\mu_{gz}}, \quad \theta = \tan^{-1} \frac{-\mu_{gx}}{\sqrt{\mu_{gy}^2 + \mu_{gz}^2}} \quad (9)$$

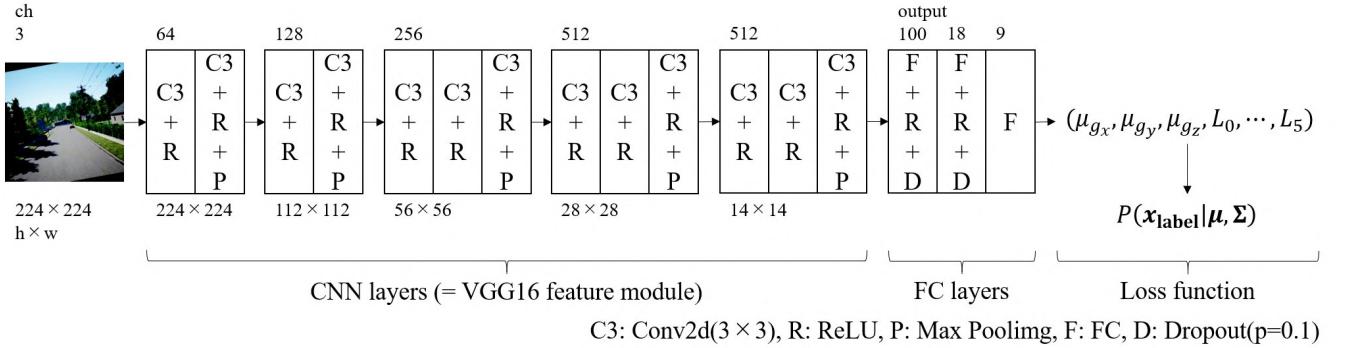


Fig. 4: Network architecture. It consists of CNN layers and FC layers. The input data is a resized image, and the output data are a mean vector and a covariance matrix. They are computed with an output from the final layer as Eq.2, 3, respectively. Log-probability of multivariate normal distribution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

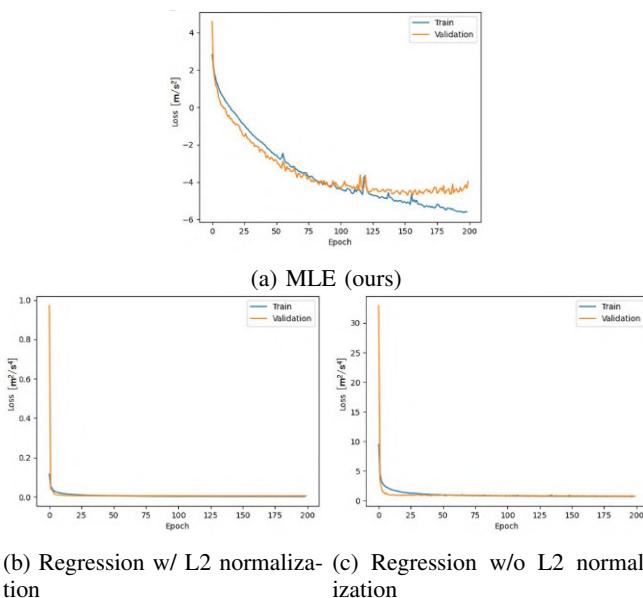


Fig. 5: Loss plotting. It is noted a loss functions of the MLE model and one of the regression models are difference. Therefore, their values can not be simply compared.

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  $\beta < TH_\beta = 0.00008814 m^3/s^6$  were selected from 1000 validation samples in “MLE (ours, selected)”.

Comparing “MLE (ours, all)” and “MLE (ours, selected)”, filtering with  $TH_\beta$  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  $\beta$  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 informa-

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.

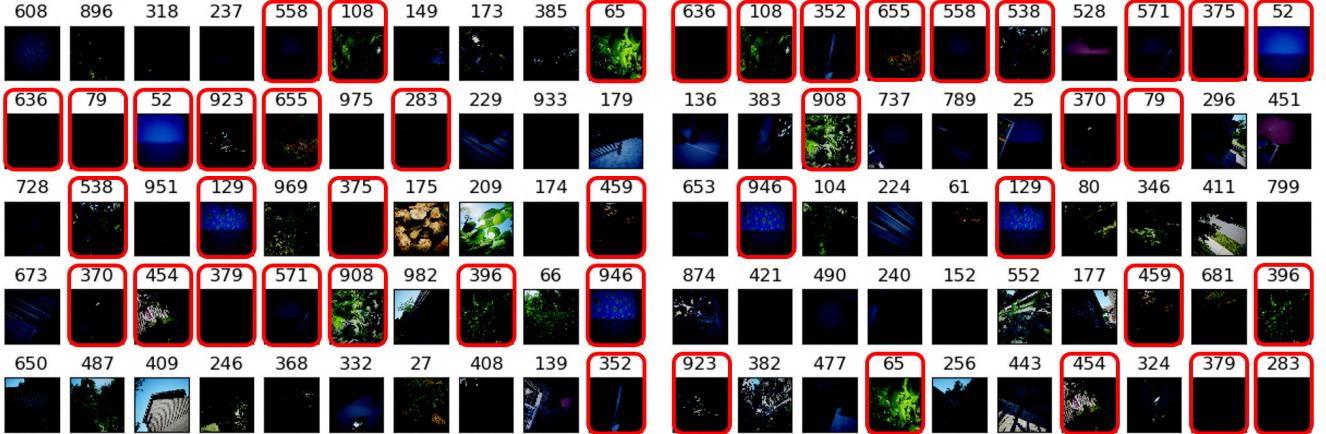
	Roll [deg <sup>2</sup> ]	Pitch [deg <sup>2</sup> ]
MLE (ours, all)	23.139	18.348
MLE (ours, selected)	9.657	4.553
Regression w/ L2 normalization	25.161	21.996
Regression w/o L2 normalization	21.668	18.213

tion. 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  $\beta$  are not complete, but many samples with large error were detected by sorting samples with  $\beta$ . A good example with large  $\beta$  and one with small  $\beta$  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  $\beta$ .

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  $\beta$  in “MLE (ours)”

Fig. 6: Sorted samples. A number above each image is an 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  $\beta$  in “MLE (ours)”. Mutual samples of the both groups are marked with red rectangles.



(a) Large  $\beta$  example

(b) Small  $\beta$  example

Fig. 7: An example with large  $\beta$  and an example with small  $\beta$ . 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](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](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 filtering 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.

# 風景知識を学習するカメラ-LiDAR-DNNによる自己姿勢推定

尾崎 亮太<sup>\*1</sup>, 黒田 洋司<sup>\*2</sup>

## Camera-LiDAR-DNN-based self-attitude estimation with learning landscape regularities

Ryota OZAKI<sup>\*1</sup> and Yoji KURODA<sup>\*2</sup>

<sup>\*1\*2</sup> Graduate School of Science and Technology, Meiji University  
1-1-1, Higashimita, Tama-ku, Kanagawa 214-8571, Japan

This paper presents camera-LiDAR-DNN (deep neural network) -based self-attitude estimation with learning landscape regularities. The proposed DNN infers a gravity direction with a camera and a LiDAR. A color image and a depth image are input to the network. The depth image is obtained with the LiDAR. Convolution layers are separately applied to them, and their outputs are concatenated. The concatenated feature is input to fully connected layers, and the gravity vector is output. The DNN is pre-trained with datasets collected in a simulator. Fine-tuning with datasets collected with real sensors is done after the pre-training. Comparison with conventional methods is carried out for the performance evaluation. Real-time estimation using EKF (extended Kalman filter) is also performed.

**Key Words :** Self-attitude estimation, Mobile robotics, Deep learning, Extended Kalman filter

### 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。関連研究<sup>(1)</sup>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。この手法はカメラを用いる手法なので、費用対効果は高いが、夜間やテクスチャのない環境では上手く機能しない。上記の問題を解決するために、本論文では、カメラとLIDARの両方を用いるDNNで重力方向を推論する。さらに、これら複数のセンサを用いることで、適用可能な環境を増やすだけでなく、より高精度かつロバストに推定を行うことを目指す。

### 2. 重力方向を推論するカメラ-LiDAR-DNN

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。

**2.1 データセット収集** データセットは、カメラ画像、LiDAR点群、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。シミュレーションデータセットはAirSim<sup>(2)</sup>で収集される。実データは、

IMU(Xsens MTi-30)とカメラ(RealSense D435)を搭載したセンサースイートを手で持って収集される。データは、センサースイートが静止しているときのみ記録される。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。

**2.2 データ前処理** 各入力データとラベルデータには前処理が適用される。カメラ画像には、リサイズ、正規化( $\text{mean} = (0.5, 0.5, 0.5)$ ,  $\text{std} = (0.5, 0.5, 0.5)$ )が適用される。点群データはデプス画像に変換される。ラベルの重力ベクトルにはL2正規化が適用される。さらに、学習時には毎エポックで、50%の確率で各データを水平反転することで、データ水増しを行う。

**2.3 ネットワーク構造** ネットワーク構造をFig.1に示す。DNNは2つのCNN層とFC層で構成されている。カラー画像とデプス画像に対して、それぞれCNN層が適用される。ただし、カラー画像に適用される層には、ImageNet<sup>(3)</sup>で事前学習されたVGG16<sup>(4)</sup>を採用する。2つのCNN層から出力される特徴量は結合され、FC層に入力される。最終層の出力は3次元の重力ベクトルである。

**2.4 最適化** 損失関数は、ラベルデータと推定値のMSE(平均二乗誤差)である。この関数で計算される損失を最小化するように、Adam(adaptative moment

<sup>\*1</sup> 明治大学理工学研究科機械工学専攻(〒214-8571 神奈川県川崎市多摩区東三田 1-1-1-D103) ce192021@meiji.ac.jp

<sup>\*2</sup> 明治大学理工学研究科機械工学専攻 ykuroda@meiji.ac.jp

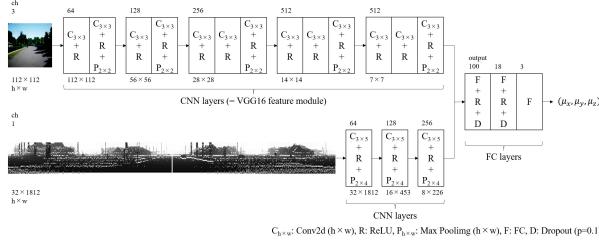


Fig. 1 Proposed network architecture.

Table 1 MAE of estimation on real data.

Method	Angle [deg]	Dataset#		
		1	2	3
Camera-LiDAR DNN (ours)	Roll	2.07	<b>2.57</b>	<b>3.49</b>
	Pitch	1.75	<b>2.86</b>	<b>2.88</b>
Camera DNN	Roll	<b>2.01</b>	3.53	5.67
	Pitch	<b>1.69</b>	3.46	4.84
LiDAR DNN	Roll	2.41	4.36	4.85
	Pitch	1.98	3.48	3.53
Statistics	Roll	15.87	15.29	19.95
	Pitch	14.81	13.42	15.91

estimation)<sup>(5)</sup>を用いてネットワークパラメータを最適化する。

### 3. 評 價 実 験

提案するDNNを評価するため、静的検証と動的検証を行い、他手法と比較する。静的検証では、訓練データでDNNを訓練し、テストデータで評価する。動的検証では、角速度センサとDNNをEKFで統合し、リアルタイム推定を評価する。

**3.1 静的検証** 提案DNNは、10000個のシミュレーションデータサンプルを用いて事前学習し、1941個の実データサンプル(Dataset#1)を用いてファインチューニングされた。バッチサイズを100、エポック数を100とした。さらに、別のエリアで、日中に収集した443個の実データサンプル(Dataset#2)と、夜間に収集した447個の実データサンプル(Dataset#3)をテストデータとして用意した。

各データセットに対する推定誤差をTable 1に示す。ここで、「Camera DNN」はカメラのみを用いたDNN<sup>(1)</sup>、「LiDAR DNN」はLiDARのみを用いたDNN、「Statistics」はラベルデータの標準偏差を表す。テストデータ(Dataset#2, #3)において、提案DNNによる推定誤差が最も小さくなかった。夜間のデータ(Dataset#3)に着目すると、日中のデータ(Dataset#2)に比べて、「Camera DNN」による推定誤差は大きくなかった。

Table 2 Error of estimated attitude at last pose.

	Roll [deg]	Pitch [deg]
Gyro+DNN (ours)	<b>-2.15</b>	<b>0.46</b>
Gyro	5.70	-5.26

**3.2 動的検証** 屋外環境で約5分間、センサー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。実験の開始時と終了時に、平らな床面にセンサを置き、そのときの真値を $\phi_{gt} = 0 \text{ deg}$ ,  $\theta_{gt} = 0 \text{ deg}$ と仮定した。i7-6700 CPU, GTX1080 GPU, 16 GB RAMを搭載したコンピュータで、DNNの推論計算時間は0.01-0.02秒だった。

表2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。

### 4. 結 言

重力方向と風景や形状の規則性を学習するDNNを用いた自己姿勢推定法を提案した。提案したDNNは、カメラ画像とLiDARデータを入力として、そのときの重力ベクトルを推論する。静的姿勢推定の実験では、提案DNNは、カメラとLiDARの両方を用いることで、昼夜問わず、より高精度に姿勢を推定できることが示された。動的姿勢推定の実験では、提案DNNがリアルタイム推定にも適用可能であることが示された。

### 参 考 文 献

- 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), pp.5557–5563.
- S. Shah, D. Dey, C. Lovett and A. Kapoor, “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles”, *Field and Service Robotics*, (2018), pp.621–635.
- 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)*, (2009), pp. 248–255.
- K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, *arXiv preprint arXiv:1409.1556*, (2014).
- D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*, (2015).

## 建造物の壁に対する相対姿勢を用いた 6DoF 位置姿勢推定

### 6DoF Localization with Relative Poses to Walls of Buildings

○学 尾崎 亮太<sup>\*1</sup>, ○正 黒田 洋司<sup>\*1</sup>

Ryota OZAKI<sup>\*1</sup>, Yoji KURODA<sup>\*1</sup>,

<sup>\*1</sup> 明治大学 Meiji University

**Key Words :** Pose estimation, Mobile robot, Artificial environment, Gauss map, Extended Kalman filter

### 1. 緒 言

移動ロボットのナビゲーションや姿勢制御を行うためには、3次元空間における時々刻々の姿勢を推定する必要がある。移動体の姿勢推定法として、内界センサを使ったデッドレコニングによる推定と、外界センサを使ったSLAMによる推定とを統合する手法が提案されている<sup>(1)</sup>。このような手法は、初期姿勢に対する相対変化を推定するため、蓄積誤差を補正することが難しい。

本研究では、従来用いられる慣性センサとSLAMの統合に加え、鉛直に建てられた建造物の壁に対する相対姿勢をワールド座標系における絶対姿勢として観測することで、誤差の蓄積を適時補正する姿勢推定法を提案する。主に、時々刻々での推定精度を重視しており、ループクローズによる過去の軌跡の補正是本研究の趣旨と一致しない。なお、本手法は、一般的な建造物の壁がおおよそ鉛直に建てられていることを利用し、事前環境地図を必要としない。屋外での実機走行実験によって本手法の有用性を示す。

座標系を以下に定義する。

#### (1) ロボット座標系

ロボットに固定され、進行方向をX軸の正とする右手直交座標系とする。各軸をX<sub>r</sub>, Y<sub>r</sub>, Z<sub>r</sub>軸とする。

#### (2) ワールド座標系

初期姿勢位置において原点がロボット座標系と一致し、重力方向をZ軸の負とする右手直交座標系とする。各軸をX<sub>w</sub>, Y<sub>w</sub>, Z<sub>w</sub>軸とする。

### 2. 建造物の壁に対する相対姿勢を用いた 6DoF 位置姿勢推定

本提案手法では、ロボットが静止している状態でIMUによって初期姿勢を求め、それに対して角速度センサを用いたデッドレコニングおよびSLAMで推定される相対姿勢変化を積算する。そして、壁面を観測した場合は、第2・1, 2・2節のように推定したロボット座標系での重力ベクトルを用いて、第2・3節のように蓄積誤差を補正する。なお、デッドレコニングで推定する姿勢を予測、SLAMで推定する姿勢を観測1、壁面に対する相対姿勢を観測2とし、拡張カルマンフィルタで統合する。車輪型ロボットで人工環境を走行することを想定しており、各姿勢角度変化が急激なものではなく、カルマンフィルタで十分扱えると仮定する。

#### 2・1 主法線抽出

測距センサで得られる点群を用いて、各注目点の近傍局所点群に主成分分析を適用することで法線ベクトルを算出する。近傍点群は、kd-tree<sup>(2)</sup>を用いて指定半径以内の点を探索することで得られる。センサから得られる点群は、センサから遠くなるほど点の密度が粗になるため、注目点とセンサとの距離が大きいほど、探索半径を大きくする。本論文内の実験では経験的に、探索半径をセンサと注目点との距離の0.09倍と設定する。

この法線群から、信頼性の高い鉛直面を持つ法線を以下の条件で抽出する。

#### (1) 注目点が持つ近傍点の数が十分多い（密度が高い）。

#### (2) 式(1)で算出される角度βが十分小さい。これは1ステップ前の推定重力ベクトルG'robotをもとに、観測された法線Nが鉛直面を持つかを判定するものである。

$$\beta = \left| \cos^{-1} \frac{\mathbf{N} \cdot \mathbf{G}'_{\text{robot}}}{\|\mathbf{N}\| \|\mathbf{G}'_{\text{robot}}\|} - \frac{\pi}{2} \right| \quad (1)$$

(3) 法線に垂直な平面と近傍点群の二乗誤差が十分小さい（平面度が高い）。

抽出された法線群でガウス球<sup>(3)</sup>を生成し、球内の点群に対してユークリッド距離に基づくクラスタ分析を適用する。ガウス球とは、各点が保有する法線ベクトルの各成分をユークリッド空間内に再配置することで得られる点群であり、この操作は一般にガウス写像と呼ばれる。なお本手法では、適当な法線を反転させガウス球をさらに半球のみに反映している。つまり、平行で向かい合う平面は同じ法線ベクトルを持つようになる。クラスタリング終了後に、メンバが閾値より少ないクラスタは外れ値として除去する。各クラスタが持つ点群の重心を算出し、それらの位置ベクトルを主法線（支配的な法線）として用いる。

## 2・2 重力ベクトル推定

クラスタリングされた主法線（ガウス球内の点）を用いてロボット座標系における単位重力ベクトル $\mathbf{G}_{\text{robot}}$ を推定する。主法線の数に応じて以下のように算出する。ただし、以下のように外積などで算出されるベクトルが、本来の重力ベクトルと方向が逆になる場合もある。そのため、算出後に、1ステップ前の重力方向をもとに、算出したベクトルの向きを判定する必要がある。

(1) 主法線の数：3以上

主法線をガウス球における点として扱い、この点群に対する主成分分析で得た法線を重力ベクトルとする。ただし、各クラスタが持つ点の数で重み付けを行う。

(2) 主法線の数：2

2本の主法線ベクトルの外積を重力方向ベクトルとする。図1(a)に推定の様子を示す。

(3) 主法線の数：1

ワールド座標系における絶対的な姿勢を得るには、平行でない鉛直面が2面以上必要であるが、1ステップ前の推定重力ベクトル $\mathbf{G}'_{\text{robot}}$ を用いて、式(2)のように部分的に推定重力ベクトルを補正し算出する。観測された主法線ベクトルを $\mathbf{N}$ とする。図1(b)に推定の様子を示す。

$$\mathbf{G}_{\text{robot}} = \frac{\mathbf{G}'_{\text{robot}} - (\mathbf{G}'_{\text{robot}} \cdot \mathbf{N})\mathbf{N}}{\|\mathbf{G}'_{\text{robot}} - (\mathbf{G}'_{\text{robot}} \cdot \mathbf{N})\mathbf{N}\|} \quad (2)$$

(4) 主法線の数：0

重力方向を推定することが出来ない。

## 2・3 推定姿勢補正

重力ベクトルの推定誤差が、推定姿勢の誤差であることを用いて補正する。なお計算は四元数（クオータニオン）を用いて行う。補正後の姿勢を表す四元数 $\mathbf{q}_{\text{pose.wall}}$ の算出を以下に示す。1ステップ前の推定重力ベクトル $\mathbf{G}'_{\text{robot}}$ と、第2・2項で推定した推定重力ベクトル $\mathbf{G}_{\text{robot}}$ において、この2ベクトル間の回転を表す四元数 $\mathbf{q}_{\text{error}}$ は、

$$\mathbf{q}_{\text{error}} = (q_{\text{error},x} \ q_{\text{error},y} \ q_{\text{error},z} \ q_{\text{error},w}) = \left( C_{X_r} \sin \frac{\gamma}{2} \ C_{Y_r} \sin \frac{\gamma}{2} \ C_{Z_r} \sin \frac{\gamma}{2} \ \cos \frac{\gamma}{2} \right) \quad (3)$$

$$\gamma = \cos^{-1} \frac{\mathbf{G}'_{\text{robot}} \cdot \mathbf{G}_{\text{robot}}}{\|\mathbf{G}'_{\text{robot}}\| \|\mathbf{G}_{\text{robot}}\|} \quad (4)$$

$$\mathbf{C} = (C_{X_r} \ C_{Y_r} \ C_{Z_r})^T = \frac{\mathbf{G}'_{\text{robot}} \times \mathbf{G}_{\text{robot}}}{\|\mathbf{G}'_{\text{robot}}\| \|\mathbf{G}_{\text{robot}}\|} \quad (5)$$

推定重力ベクトル $\mathbf{G}'_{\text{robot}}$ に対応する推定姿勢を表す四元数を $\mathbf{q}'_{\text{pose}}$ としたとき、次のように補正される。

$$\mathbf{q}_{\text{pose.wall}} = \mathbf{q}'_{\text{pose}} \mathbf{q}_{\text{error}} \quad (6)$$

$\mathbf{q}_{\text{pose.wall}}$ および $\mathbf{q}'_{\text{pose}}$ はワールド座標系での姿勢であり、 $\mathbf{q}_{\text{error}}$ はロボット座標系での誤差（回転）である。

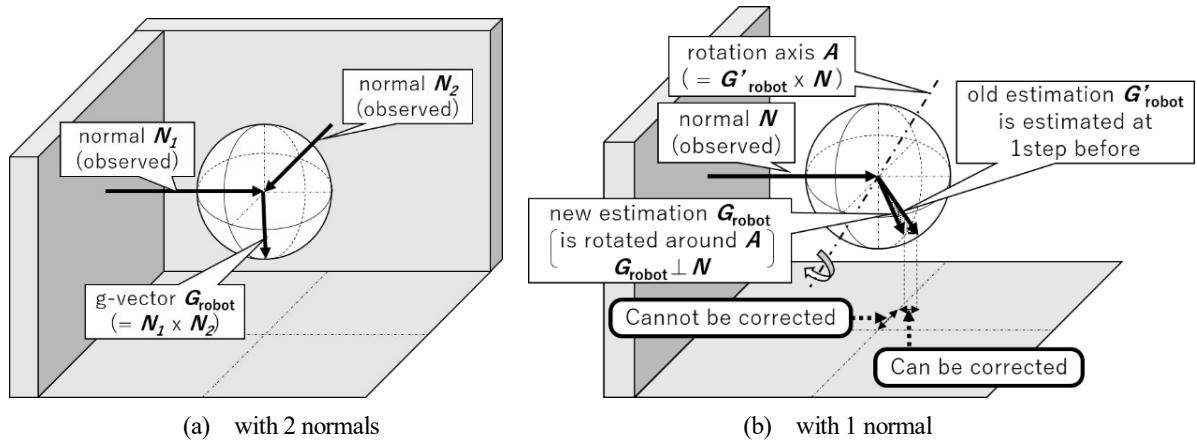


Fig. 1 Estimation and correction of the gravity vector.

### 3. 評価実験

#### 3・1 実験概要

起伏のあるコースで移動ロボットを走行させ、提案手法および比較手法で、時々刻々のロボットの自己姿勢を推定する。走行中の推定姿勢を評価することは難しいため、推定姿勢と並進速度を統合することで、走行中の推定姿勢誤差を、走行終了時の推定位置誤差に置き換えて評価する。つまり、同じ並進速度を用いた場合、姿勢の推定誤差が小さいほど、推定位置（軌跡）の誤差が小さくなると仮定する。

#### 3・2 実験環境

本実験での提案手法のシステム図を図2に示す。図中、Staticはロボットが静止している状態での初期姿勢推定のシステム、Dynamicは走行開始後のシステムをそれぞれ表している。カルマンフィルタの観測で用いるSLAMとして、LSD-SLAM<sup>(4)</sup>を用いる。LSD-SLAMとは単眼カメラSLAMである。推定のレートの速さとスケールに依存しないことが姿勢推定に適していると判断し選定する。並進速度は、ホイールオドメトリで得られる並進速度を用いる。姿勢推定の比較手法として、ジャイロスコープによるデッドレコニングと、単独のLSD-SLAMを用いる。提案手法と同様に、これらの手法で姿勢をそれぞれ推定し、ホイールエンコーダの並進速度と統合する。なお、本研究の趣旨より、LSD-SLAMでのループクローズは行わない。

ロボットに搭載し、使用するセンサは、Velodyne HDL-32E、Xsens MTi30、RealSense D435、ホイールエンコーダである。RealSense D435はRGB-Dカメラであるが、LSD-SLAMのために画像のみを利用しているため、深度情報は利用していない。評価に用いる真値として、走行開始時と走行終了時における、モーションキャプチャで得られる位置姿勢を用いる。本実験ではVicon Vero v1.3Xを用いる。図3に、明治大学生田キャンパス内の実験環境の航空写真を示す。走行コースは、1周約250m、高低差約3mであり、図3に示すように3周走行する。

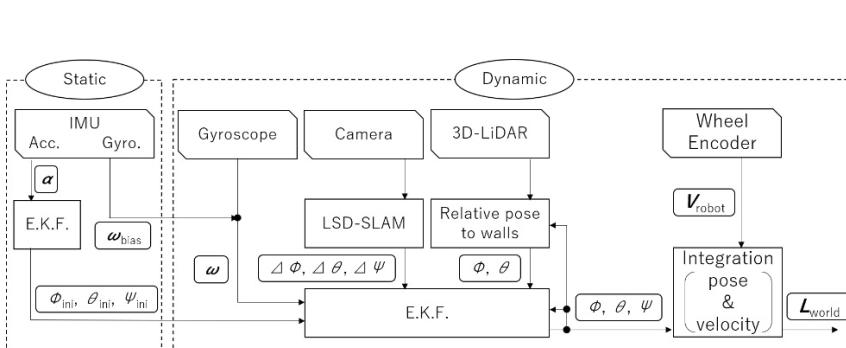


Fig. 2 System configuration diagram of the proposed method.

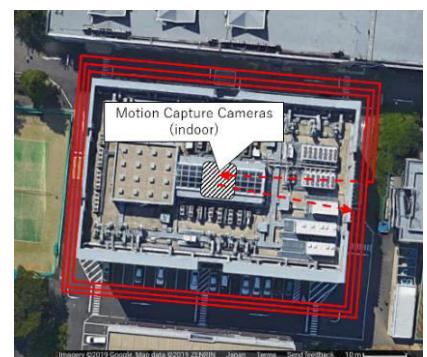


Fig. 3 The experimental environment.

### 3・3 実験結果

表1に、各手法での推定位置姿勢の誤差を示す。表中の誤差は、ワールド座標系の座標軸を基準とした位置および姿勢角の誤差である。図4に、各手法での推定されたロボットの軌跡を示す。ユークリッド距離で表した位置推定誤差に関して、比較手法に比べて、提案手法の値が小さいことから、走行中の姿勢推定の誤差も提案手法の方が小さいと言える。特に、 $Z_w$ 軸（鉛直）方向の並進誤差が小さく抑えられており、壁面を用いたロール、ピッチ補正による結果だと考察する。

Table 1 Errors in estimated pose and position

error in ...	$X_w$ [m]	$Y_w$ [m]	$Z_w$ [m]	Euc.dist. [m]	$\text{roll}_w$ [deg]	$\text{pitch}_w$ [deg]	$\text{yaw}_w$ [deg]
proposed method	+1.8843	-0.6136	+1.9233	+2.762	-2.069	-0.095	-0.300
LSD-SLAM	+4.8471	-2.3376	+9.0677	+10.544	+6.922	-8.721	+18.257
Gyrodometry	+2.0761	-0.2382	-2.6726	+3.393	-2.984	+6.970	-0.963

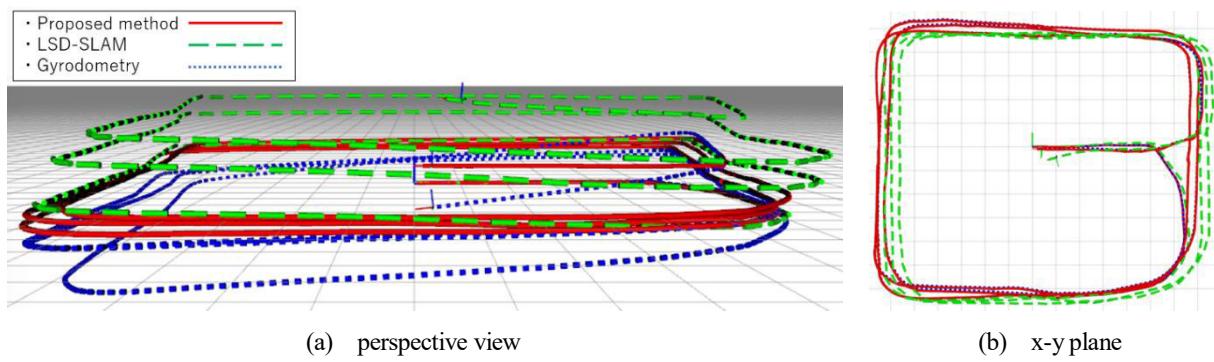


Fig. 4 Estimated trajectories of the robot in the experiment.

### 4. 結 語

従来の慣性センサとSLAMによる移動ロボットの姿勢推定に加え、鉛直に建てられている建造物の壁に対する相対姿勢を利用する姿勢推定法を提案した。実験結果より、本提案手法の、蓄積誤差に対する補正の有用性が示された。

### 謝 辞

本研究の一部は、国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の、次世代人工知能・ロボット中核技術開発事業による支援を受けた。ここに篤く御礼申し上げる。

### 文 献

- (1) Stumberg, L.von, Usenko, V. and Cremers, D., “Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization”, *in International Conference on Robotics and Automation* (2018), pp.2510–2517.
- (2) Bentley, J. L., “Multidimensional binary search trees used for associative searching”, *Communications of the ACM*, Vol.18, No.9 (1975), pp.509–517.
- (3) Horn, B. K. P., “Extended gaussian images”, *Proceedings of the IEEE*, Vol.72, No.12 (1984), pp.1671–1686.
- (4) Engel, J., Stueckler, J. and Cremers, D., “LSD-SLAM: Large-Scale Direct Monocular SLAM”, *European Conference on Computer Vision* (2014), pp.834–849.

# 鉛直壁面を用いた移動ロボットのための自己姿勢推定

Pose Estimation for mobile robots with vertical walls

○学 尾崎亮太 (明治大) 正 黒田洋司 (明治大)

Ryota OZAKI, Meiji University, ee53031@meiji.ac.jp

Yoji KURODA, Meiji University

This paper presents a pose estimation method for mobile robots with vertical walls in urban areas. This proposed method exploits the fact that most of all artificial walls are built vertically. It corrects accumulative error of estimated poses by estimating the gravity vector with the fact without any 3d maps or 3d models. The poses which are estimated with vertical walls are integrated with integration of angle velocity from inertial sensors thorough extended Kalman filter. To evaluate the proposed method, outdoor experiments with an actual robot are performed. It shows the method keeps correcting accumulative error while the robot moves.

**Key Words:** Mobile robot, Pose Estimation, 3D point cloud, Gauss map, Artificial environment

## 1 緒言

移動ロボットのナビゲーションや姿勢制御を行うためには、3次元空間における時々刻々の姿勢を推定する必要がある。移動体の姿勢は、一般に、ジャイロスコープや加速度センサを使って推定することが多い。しかし、地面上を走行する移動ロボットの場合、ロボット自身の加速度や地面からの振動が推定精度に影響を与える[1]。また、ジャイロスコープを用いた角速度の積分(レート積分)では、積算誤差が発生する。一方、この蓄積誤差を補正するため、事前に用意した情報とセンサ情報をマッチングする手法があり、特に事前環境地図を用いる手法が多く提案されている[2]。しかしながら、事前地図の使用は、適用可能環境を限定する。

本研究では、従来のレート積分による姿勢推定に対して、鉛直な壁面を用いてロボット座標系での重力方向ベクトルを推定し補正する。提案手法は、一般的な建造物の壁がおおよそ鉛直に建てられていることを利用し、事前環境地図を必要とせずに蓄積誤差を補正する。屋外での実機走行実験によって、本研究の有用性を示す。

## 2 鉛直壁面を用いた姿勢推定

本手法では、ロボットが静止している状態で慣性センサによって初期姿勢を求め、それに対してレート積分で推定される相対姿勢変化を積算する。そして、壁面を観測した場合は、ロボット座標系での重力ベクトルを推定し、蓄積誤差を補正する。重力ベクトルの推定及び補正の詳細を2.1～2.3節で記述する。レート積分で推定する姿勢を予測、壁面情報で推定する姿勢を観測とし、拡張カルマンフィルタ(E.K.F.)で統合する。車輪型ロボットで人工環境を走行することを想定しており、各姿勢角度変化が急激なものではなく、E.K.F.で十分扱えると仮定する。

座標系を以下に定義する。

- ロボット座標系：ロボットに固定され、進行方向をx軸の正とする右手直交座標系とする。
- ワールド座標系：ロボット初期姿勢位置において原点がロボット座標系と一致し、重力方向をz軸の負とする右手直交座標系とする。

### 2.1 主法線抽出

3次元距離センサで得られる点群情報に対して主成分分析を用いることで、各注目点に対する法線を算出する。この法線群から、信頼性の高い鉛直面を持つ法線を以下の条件で抽出する。

- 注目点が持つ近傍点の数が十分多い（密度が高い）。
- 1ステップ前の推定重力ベクトル $\mathbf{G}'_{\text{robot}}$ をもとに、観測された法線 $\mathbf{N}$ が鉛直面を持つかを判定する（式(1)で算出される角度 $\beta$ が十分小さい）。

$$\beta = \left| \cos^{-1} \frac{\mathbf{N} \cdot \mathbf{G}'_{\text{robot}}}{\|\mathbf{N}\| \|\mathbf{G}'_{\text{robot}}\|} - \frac{\pi}{2} \right| \quad (1)$$

- 法線に垂直な平面と近傍点群の二乗誤差が十分小さい（平面度が高い）。

抽出された法線群をガウス球[3]に写像し、球内の点群に対してユークリッド距離に基づくクラスタ分析を適用する。本手法では、適当な法線を反転させガウス球を半球のみに反映する。つまり、平行で向かい合う壁面は同じ法線ベクトルを持つようになる。クラスタリング終了後に、メンバが閾値より少ないクラスタは外れ値として除去する。各クラスタが持つ点群の重心を算出し、それらの位置ベクトルを支配的な法線（主法線）として用いる。

### 2.2 単位重力ベクトル推定

クラスタリングされた主法線を用いてロボット座標系における単位重力ベクトル $\mathbf{G}'_{\text{robot}}$ を推定する。主法線の数に応じて以下のように算出する。

- 主法線の数：3以上  
主法線をガウス球における点として扱い、この点群に対する主成分分析で得た法線を重力ベクトルとする。ただし、各クラスタが持つ点の数で重み付けを行う。
- 主法線の数：2  
2つの法線ベクトルの外積を重力方向ベクトルとする。
- 主法線の数：1  
ワールド座標系における絶対的な姿勢を得るには、平行でない鉛直面が2面以上必要であるが、1ステップ前の推定重力ベクトル $\mathbf{G}'_{\text{robot}}$ を用いて、式(2)のように部分的に推定重力ベクトルを補正し算出する。観測された主法線ベクトルを $\mathbf{N}$ とする。

$$\mathbf{G}'_{\text{robot}} = \frac{\mathbf{G}'_{\text{robot}} - (\mathbf{G}'_{\text{robot}} \cdot \mathbf{N})\mathbf{N}}{\|\mathbf{G}'_{\text{robot}} - (\mathbf{G}'_{\text{robot}} \cdot \mathbf{N})\mathbf{N}\|} \quad (2)$$

- 主法線の数：0

重力方向を推定することが出来ない。

### 2.3 姿勢推定補正

重力ベクトルの推定誤差が、推定姿勢の誤差であることを用いて推定を補正する。補正後の姿勢を表す四元数 $\mathbf{q}_{\text{pose,wall}}$ の算出法を以下で示す。1ステップ前の推定重力ベクトル $\mathbf{G}'_{\text{robot}}$ と、2.2節で推定した推定重力ベクトル $\mathbf{G}'_{\text{robot}}$ において、この2ベクトル間の回転を表す四元数を $\mathbf{q}_{\text{error}}$ とする。推定重力ベクトル $\mathbf{G}'_{\text{robot}}$ に対応する推定姿勢を表す四元数を $\mathbf{q}'_{\text{pose}}$ としたとき、次のように推定姿勢は補正される。

$$\mathbf{q}_{\text{pose,wall}} = \mathbf{q}'_{\text{pose}} \mathbf{q}_{\text{error}} \quad (3)$$

$\mathbf{q}_{\text{pose,wall}}$ および $\mathbf{q}'_{\text{pose}}$ はワールド座標系での姿勢であり、 $\mathbf{q}_{\text{error}}$ はロボット座標系での誤差（回転）である。

**Table 1:** Errors in pose estimations

	error in...	roll <sub>w</sub> [deg]	pitch <sub>w</sub> [deg]	yaw <sub>w</sub> [deg]
1 round	proposed method	-0.947	+1.381	+3.250
	rate-integration	-0.287	+2.206	+3.154
	AHRS	-1.000	+2.294	-2.918
2 rounds	proposed method	-1.048	+1.872	+5.105
	rate-integration	-3.139	+0.167	+5.052
	AHRS	-1.218	+1.832	+26.718
3 rounds	proposed method	-0.714	+1.517	+9.554
	rate-integration	-2.070	+3.259	+9.200
	AHRS	-1.198	+1.663	+19.476
(absolute value)	proposed method	0.903	1.590	5.970
	rate-integration	1.832	1.878	5.802
	AHRS	1.139	1.930	16.371
cov.	proposed method	0.020	0.043	6.998
	rate-integration	1.384	1.647	6.373
	AHRS	0.010	0.071	159.139

### 3 評価実験

#### 3.1 実験概要

起伏のあるコースで移動ロボットを走行させ、提案手法および従来手法により、時々刻々のロボットの自己姿勢を推定する実験を行った。そして、走行距離の変化に応じた姿勢推定の誤差を評価した。本実験では、従来手法として、レート積分で推定された姿勢と姿勢計測装置（AHRS）で計測された姿勢を用いた。真値には、走行開始時と終了時にモーションキャプチャ（Vicon Vero v1.3X）で計測した姿勢を用いた。走行コースは、屋外の1周約250[m]、高低差約3[m]のコースであり、周回数を変えて複数の走行を行った。ただしスタート地点とゴール地点はモーションキャプチャを設置した屋内である。ロボットに搭載したセンサは、Velodyne HDL-32E、Xsens MTi30、ホイールエンコーダである。

#### 3.2 実験結果

表1に姿勢推定の誤差を示す。表中の誤差は、ワールド座標系の座標軸を基準とした姿勢角の誤差である。レート積分に対して提案手法は、平均的にロール角、ピッチ角の推定誤差が小さく、走行距離の変化における推定誤差の分散も小さい。AHRSによる姿勢計測に関して、計測は走行終了時にロボットが静止している状態で行われており、走行中における計測に比べて精度よく計測ができるはずである。そこで、式(4)のように推定姿勢と並進速度を統合した場合、ロール角、ピッチ角の推定誤差はZ<sub>w</sub>軸方向の並進誤差として表れるという仮定のもと、別の評価を行う。

$$\begin{aligned} \mathbf{L}_{\text{world},k} &= \mathbf{L}_{\text{world},k-1} + \mathbf{R}_k^{-1} \mathbf{V}_{\text{robot},k} dt \\ &= \mathbf{L}_{\text{world},k-1} + \mathbf{R}_k^T \mathbf{V}_{\text{robot},k} dt \\ &= \begin{pmatrix} L_{X_w,k-1} \\ L_{Y_w,k-1} \\ L_{Z_w,k-1} \end{pmatrix} + \mathbf{R}_k^T \begin{pmatrix} V_{\text{robot},X_r,k} \\ 0 \\ 0 \end{pmatrix} dt \end{aligned} \quad (4)$$

$\mathbf{V}_{\text{robot}}$ はロボット座標系での並進速度ベクトル、 $\mathbf{L}_{\text{world}}$ はワールド座標系でのロボットの推定位置ベクトルである。本実験では、ホイールエンコーダで計測される並進速度を用いる。表2に、推定姿勢と並進速度を統合した場合のZ<sub>w</sub>軸方向の並進誤差を示す。提案手法の並進誤差が比較手法の誤差より小さいことから、走行中の姿勢角推定の誤差も小さいと言える。

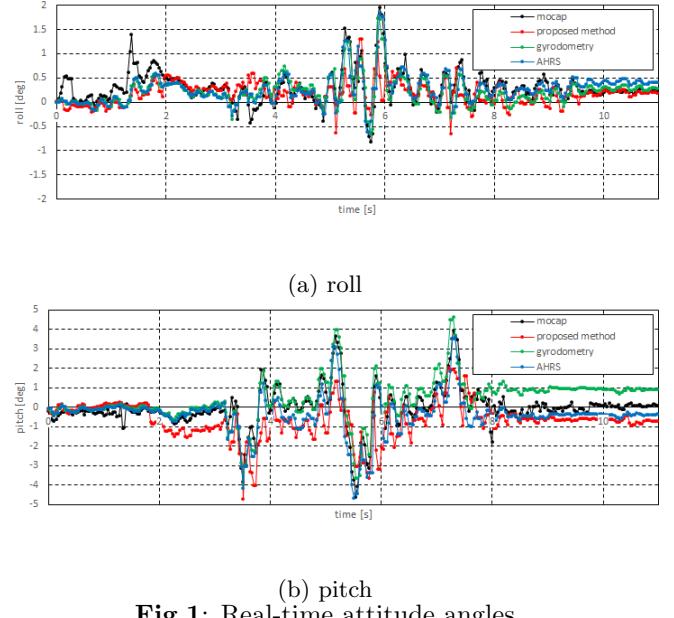
#### 3.3 追加実験

提案手法の瞬間的な推定精度を検証するため、移動ロボットが走行時に振動するような玄関マットの上を走行させて、姿勢を推定する実験を行った。モーションキャプチャで計測される時々刻々の姿勢を真値として評価した。

図1に、推定姿勢角および真値の時間推移結果を示す。真値に対する推定誤差の絶対値の時間平均は、ロール角に関して、提案手法で6.877deg/s、レート積分で4.805deg/s、AHRSで5.003deg/s

**Table 2:** Errors in position estimations

	error in...	Z <sub>w</sub> [m]
1 round	proposed method	+0.0185
	rate-integration	-0.2194
	AHRS	+0.6763
2 rounds	proposed method	-0.1813
	rate-integration	+2.9508
	AHRS	+1.2496
3 rounds	proposed method	+0.6073
	rate-integration	+1.2231
	AHRS	+1.2725



**Fig.1:** Real-time attitude angles

であった。また、ピッチ角に関して、提案手法で25.578deg/s、レート積分で17.232deg/s、AHRSで14.024deg/sであった。比較手法に比べて、提案手法は瞬間的な精度が低い結果となった。鉛直面の抽出の処理時間や、観測ノイズが原因であると考察できる。瞬間的な推定精度が低いことが、提案手法の今後の課題であるとわかった。

### 4 結言

従来のレート積分による姿勢推定に対して、鉛直な壁面を用いてロボット座標系での重力方向ベクトルを推定し補正する手法を提案した。実験結果より、提案手法が事前環境地図を必要とせずに蓄積誤差を補正することを示した。

### 謝辞

本研究の一部は、国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の、次世代人工知能・ロボット中核技術開発事業による支援を受けた。ここに篤く御礼申し上げる。

### 参考文献

- Vaganay, J., Aldon, M.J. and Fournier, A., "Mobile robot attitude estimation by fusion of inertial data," Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp.277–282, 1993.
- 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, pp.312–328, 2007.
- Horn, B.K.P., "Extended gaussian images", Proceeding of the IEEE, vol.72, no.12, pp.1671–1686, 1984.

# 人工環境の平面をランドマークとして用いる EKF-SLAM

○尾崎 亮太 (明治大学), 黒田 洋司 (明治大学)

## EKF-SLAM with planar landmarks in artificial environments

○ Ryota OZAKI (Meiji University), and Yoji KURODA (Meiji University)

**Abstract :** This paper presents online SLAM for localization of mobile robots in artificial environments. This method exploits global planar features as landmarks in extended Kalman filter (EKF). Planes are extracted from point-cloud of 3D-LiDAR as normals. These normals are projected onto “depth-Gaussian sphere”. The state vector of EKF has states of both a robot and landmarks. Observed planes and landmark planes are associated, and the state vector is updated. To avoid mismatching between observations and landmarks, some conditions are set. To evaluate the proposed method, an experiment with an actual robot was performed. It shows the method suppresses accumulative error of localization by using the landmarks.

### 1. 緒言

移動ロボットの分野では、未知の環境を走行するとき、自己位置推定と地図生成を同時にSLAM[1]がよく用いられる。ICP[2]やNDT[3]を用いたスキャンマッチングは、典型的なSLAMの手法の1つだ。Zhang[4]は、低ドリフトと低計算量を目的として、エッジと平面を特徴量として用いるスキャンマッチングを提案した。しかしながら、これらのようなスキャンマッチングは、相対変化量を積算するので、蓄積誤差を補正するのが難しい。一方、ランドマークSLAM[1]は、ランドマークを観測している限り、蓄積誤差を補正しやすい手法である。特に、EKFを用いたランドマークSLAMはよく知られている[5]。また、カメラを用いたVisual SLAMの分野では、平面特徴量を用いた手法が提案されている[6, 7]。しかし、これらの論文では、ロボットが通り過ぎて観測できなくなったランドマークの扱いに関して明記されていない。つまり、ロボットが同じ場所に戻ってくることが想定されておらず、誤マッチングを避ける方法が明記されていない。

上記の問題を解決するため、本論文では、ロボットが同じ場所に戻ってくることも想定した、人工環境の平面をランドマークとして用いるEKF-SLAMを提案する。

### 2. 人工環境の平面をランドマークとして用いる EKF-SLAM

#### 2.1 平面特徴量の抽出

まず、3D-LiDARの点群の各局所点群に主成分分析(PCA)を適用することで、法線群を生成する。その法線群から、条件を設定することで、高い平面度を持つ平面のみを選定する。選定されたそれぞれの法線の原点を、原点に移動させて、深さを附加して法線の各成分をユーク

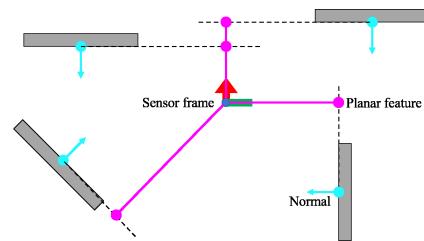


Fig. 1: Depth-Gaussian sphere

リッド空間に写像する（深さ付きガウス球[8]）。Fig.1に、深さ付きガウス球の概要図を示す。深さ付きガウス球に写像された点群に対してユークリッドクラスタ分析を適用し、メンバの数が十分多いクラスタを選定する。選定された各クラスタの重心を特徴量として抽出する。

#### 2.2 EKF の構成

状態ベクトルはロボットとランドマークの状態で構成され、EKFでそれを推定する。ジャイロスコープとホールエンコーダを用いて予測を行う。また、観測された特徴量とランドマークの対応付けで更新を行う。対応付けされなかった特徴量は新たなランドマークとして登録される。

#### 2.3 データアソシエーション

観測された特徴量とランドマークとのマハラノビス距離に閾値を設定することで、それらの対応付けを行う。ただし、幾何学的拘束を用いることで誤マッチングへの対策を行う。具体的には、1) 平面の裏側は観測できない拘束と、2) 距離が遠すぎるランドマークは観測できない拘束である。さらに、必要に応じて、ランドマークの統合を行う。つまり、ロボットが同じ場所に戻って来たときなどに、そのステップで観測しているランドマークと、過去に観測さ

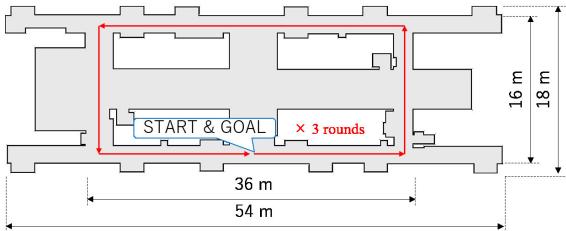


Fig. 2: Floor map

れたランドマークが同一だと判定されたら、統合するように実装されている。ただし、過去のステップで同時に観測されたランドマーク同士は、明らかに同一のランドマークでないため、統合されない。

### 3. 実験

#### 3.1 実験概要

移動ロボットを走行させて、ロボットの位置と姿勢を推定する実験を行った。その推定は、提案手法と従来手法を用いてそれぞれ行われた。ただし、走行コースの全体でロボットの位置と姿勢の真値を計測することは難しいため、走行終了時（ゴール地点）の推定値のみを評価した。つまり、コース全体で発生した蓄積誤差が、走行終了時の誤差として表れると仮定して評価した。さらに、スタート地点とゴール地点を同一にし、ロボットがスタート地点に戻ってくるように操作することで、簡易的に走行終了時の真値を  $(x, y, z, \phi, \theta, \psi) = \mathbf{0}_6$  とした。

ロボットに搭載したセンサは、3D-LiDAR (Velodyne HDL-32E)、IMU (Xsens MTi30)、ホイールエンコーダである。走行コースを Fig.2 に示す。1周 100 m のコースをロボットに 3 周走行させた。従来手法として、ジャイロオドメトリ [9] と LOAM[4] を用いた。

#### 3.2 実験結果

Table 1 に、ゴール地点での自己位置推定の誤差を示す。提案手法による推定誤差が、従来手法の誤差より小さい結果となった。コース全体で発生した蓄積誤差が、走行終了時の誤差として表れると仮定しており、3 周走行後の提案手法によるユークリッド距離の推定誤差  $d_{\text{Euc}}$  が小さいことから、提案手法が蓄積誤差を補正しながら走行できたと言える。また、ロボットがスタート地点に戻って来たとき、いくつかのランドマークは統合された。一方で、従来手法では蓄積誤差を補正するのは難しかった。特に、LOAMによる推定では、各ステップの変位の推定精度は十分だが、推定が一度ずれてしまうと、その後は修正するのが難しい。

Table 1: Error of the estimated return pose

error in...	Proposed method	Gyrodometry	LOAM
$x[\text{m}]$	+0.005	-0.632	-0.117
$y[\text{m}]$	+0.041	+0.036	+0.077
$z[\text{m}]$	-0.088	-1.212	-0.574
$d_{\text{Euc}}[\text{m}]$	0.097	1.367	0.590
$\phi[\text{deg}]$	-0.6	-2.1	-1.3
$\theta[\text{deg}]$	+0.3	+1.2	-2.7
$\psi[\text{deg}]$	-1.2	-2.7	-1.4

### 4. 結言

人工環境の平面をランドマークとして用いる EKF-SLAM を提案した。その手法では、ランドマークを観測することで、ロボットの位置、姿勢とランドマークの位置が更新される。観測された平面特徴量とランドマークの対応付けでは、誤マッチングを防ぐよういくつかの条件を設定した。また、ロボットが同じ場所に戻って来る想定し、ランドマークの統合も実装した。

### 謝辞

本研究の一部は、国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の、次世代人工知能・ロボット中核技術開発事業による支援を受けた。ここに篤く御礼申し上げる。

### 参考文献

- [1] S. Thrun, W. Burgard and D. Fox: “probabilistic robotics”, *The MIT Press*, pp.309–336(2005).
- [2] S. Rusinkiewicz and M. Levoy: “Efficient Variants of the ICP Algorithm”, *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pp.145–152(2001).
- [3] P. Biber and W. Straßer: “The Normal Distributions Transform: A New Approach to Laser Scan Matching”, *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol.3, pp.2743–2748(2003).
- [4] J. Zhang and S. Singh: “LOAM: Lidar Odometry and Mapping in Real-time”, *Proceedings of Robotics: Science and Systems Conference*, pp.161–195(2014).
- [5] S. Huang and G. Dissanayake: “Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM”, *IEEE Transactions on Robotics*, Vol.23, No.5, pp.1036–1049(2007).
- [6] J. Martinez-Carranza and A. Calway: “Unifying Planar and Point Mapping in Monocular SLAM”, *Proceedings of the British Machine Vision Conference (BMVC)*, pp.43.1–43.11(2010).
- [7] A. P. Gee, D. Chekhlov, WW. Mayol-Cuevas, A. Calway: “Discovering Planes and Collapsing the State Space in Visual SLAM”, *Proceedings of British Machine Vision Conference*, pp.1–10(2007).
- [8] 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講演予稿集, pp.453–458(2014).
- [9] J. Borenstein and L. Feng: “Gyrodometry: a new method for combining data from gyros and odometry in mobile robots”, *Proceedings of IEEE International Conference on Robotics and Automation*, pp.423–428(1996).

# Edge-node Map 及び交差点形状マッチングを用いた ナビゲーションシステムの開発

恩田知弥<sup>\*1</sup>, 大石朋孝<sup>\*1</sup>, 有馬純平<sup>\*2</sup>, 尾崎亮太<sup>\*2</sup>, 隼田駿大<sup>\*2</sup>, 黒田洋司<sup>\*2</sup>

## Development of navigation system using Edge-Node Map and intersection shape matching

Kazuya ONDA<sup>\*1</sup>, Tomotaka OISHI<sup>\*1</sup>, Jumpei ARIMA<sup>\*2</sup>,  
Ryota OZAKI<sup>\*2</sup>, Toshihiro HAYATA<sup>\*2</sup>, and Yoji KURODA<sup>\*2</sup>

<sup>\*1</sup> Graduate School of Science and Technology, Meiji University

1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa 214-8571, Japan

<sup>\*2</sup> School of Science and Technology, Meiji University

1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa 214-8571, Japan

Acquiring detailed information on the activity environment in advance is effective for performing autonomous movement. However, in order to acquire the data, it is necessary to run the entire activity environment at least once. When doing activities in a vast range, this will cost a great deal. In addition, systems that depend on such information are very vulnerable to changes in the environment because they depend on the state at the time of acquiring the data. Then, we propose a navigation system using Edge-node map created from electronic map that anyone can use. In this system, localization is performed by comparing an intersection estimated from sensor data acquired in real time with the map to realize autonomous movement independent of the pre-environment map.

**Key Words :** Autonomous navigation, Edge-node map, mobile robot

## 1. 緒 言

近年、自動運転車を中心とした自律移動ロボットの研究開発が盛んに行われている。その中でも警備ロボットや配達ロボットなどの人間に近い環境で活動するロボットへの関心が高まっている。これらのロボットが自律移動を行うためには、環境認識、自己位置推定、経路探索、コントロールの四要素が非常に重要である。現在は、自律移動ロボットが事前に取得した環境のデータから作成した事前地図を所有し、その地図をもとに自律移動を行うことが多い。しかし、都市環境では建物の新造や取り壊し、道路工事などによって事前に取得したデータと実際の環境が異なる可能性が考えられる。このような場合、事前環境地図を利用したナビゲーションシステムでは、システムの安定性は実際の環境との差異に大きく左右されてしまう。そのため、ロバスト性の高い自律ナビゲーションを行うためには、事前環境地図に依存しないシステムが求められる。そこで我々は、一般に公開されている電子地図

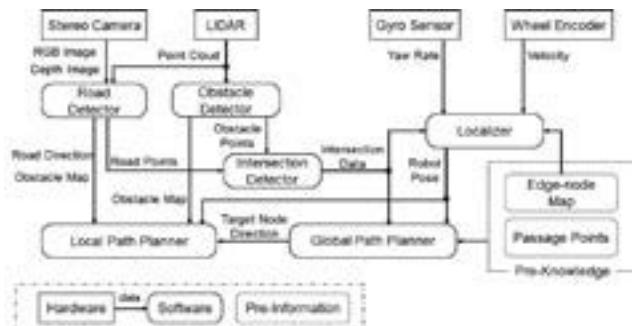


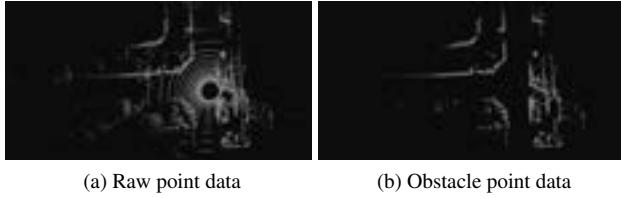
Fig. 1: System architecture

から交差点を node、道を edge とした Edge-node map を作成し、これと通過すべき node のみを事前情報としたナビゲーションシステムを提案する。

本システムでは、レーザセンサ及びステレオカメラから取得したデータから交差点形状推定を行い、事前作成した Edge-node map と比較することにより自己位置推定を行う。また、リアルタイムで取得したセンサデータから走行可能領域検出を行うことにより node 間を道なりに走行することで、周辺環境に則したナビゲーションを実現する。

<sup>\*1</sup> 明治大学大学院理工学研究科機械工学専攻（〒214-8571 神奈川県川崎市多摩区東三田1-1-1）ce182020@meiji.ac.jp

<sup>\*2</sup> 明治大学理工学部機械工学科（〒214-8571 神奈川県川崎市多摩区東三田1-1-1）ykuroda@meiji.ac.jp



(a) Raw point data                          (b) Obstacle point data

Fig. 2: Drivable area detection

## 2. システム構成

本節では、我々が開発している移動ロボット INFANT (INtegrated Foundations for Advanced Navigation Technology) の構成について述べる。INFANT は 2 横駆動の差動式、ロッカーボギー構造を有しており、その大きさは  $0.60\text{m(W)} \times 0.85\text{m(D)} \times 1.45\text{m(H)}$  である。モータ制御のための PC との通信は USB によって行う。センサは、LIDAR (HDL-32e)，ステレオカメラ (ZED)，AHRS(Xsens MTi-30)，ホイールエンコーダが搭載されている。これらのセンサ処理や自律移動に関する演算処理は、ノート PC(Intel®Corei7-6700 HQ 2.60 GHz, RAM 16GB) を 2 台使用する。

## 3. 提案手法

図 1 に我々の提案する自律ナビゲーションシステムの概略図を示す。提案手法では、事前に走行環境の形状情報を保持していないため、リアルタイムに取得したセンサデータが非常に重要である。また、ロボットの状態を常に把握し、ロボットの状態に応じて行動を変化させる必要がある。人がある地点から目的地に向かう場合を考えると、自分が今いる位置を把握したり、向かう方向を変化させたりするとき、交差点をキーポイントとすることが多い。そこで本システムでは、交差点を行動変化のキーポイントに設定し、LIDAR から取得した形状情報とステレオカメラから取得した意味情報を組み合わせることにより、交差点形状推定と走行可能領域検出を行う。これにより、Edge-node map 上での自己位置推定と交差点間の道なり走行を行い、活動環境の事前走行を必要としない自律ナビゲーションシステムを実現する。以下各節では、本システムを構成する上で重要な要素である「環境認識」、「自己位置推定」、「経路計画」の詳細を述べる。

**3.1 環境認識** 提案手法では、活動環境の形状情報を事前に取得していないため、走行時の環境認識が非常に重要な役割を担う。本システムにおいて環境認識は、走行可能領域検出と交差点形状推定の 2 要素で構成されるが、安定した自律走行を行うためにどちらも最も重要な要素のひとつと言える。

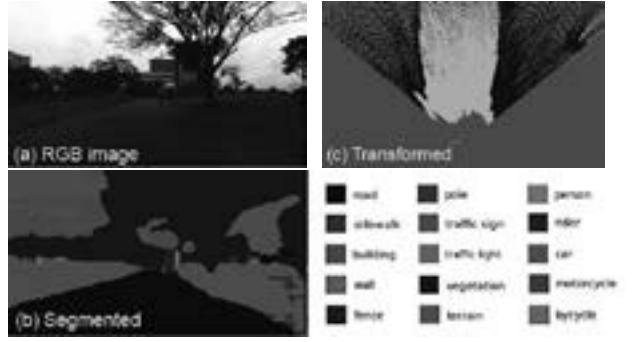


Fig. 3: An example of segmentation

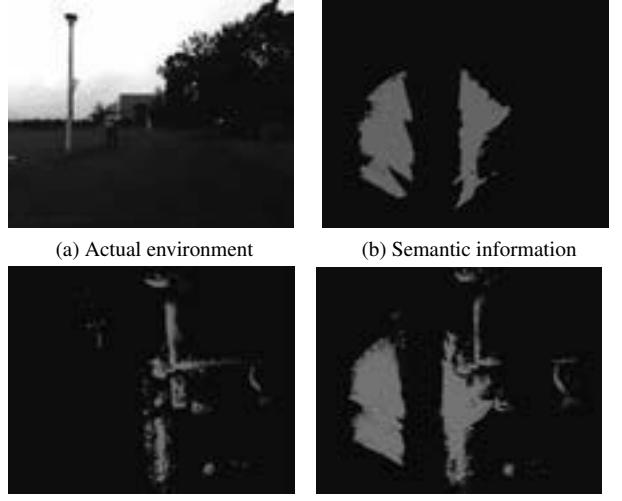


Fig. 4: Integration of shape and semantic information

**3.1.1 走行可能領域の検出** 走行可能領域の検出は、node 間の道なり走行および交差点形状推定を行うために不可欠である。走行不可能な領域は、高さ変化があり物理的に走行ができない領域と芝生などのように高さの変化はないが走行すべきではない領域に大別される。本システムでは、このどちらにも対応すべく高さのある障害物の検出には LIDAR、芝生の検出にはステレオカメラを利用し、これらのデータを統合することにより安全に走行できる領域の検出を行う。

### 形状情報による走行可能領域の検出

壁や塀など「高さ変化が大きい障害物」に関しては、従来障害物検出に用いられる手法である Min-Max 法<sup>(1)</sup>を用いて検出を行う。Min-Max 法は水平面を格子状に区切り、式 1 に示すように各格子内に含まれる点群  $P$  の最大値、最小値の差が一定の域値  $V_{dif}$  を超えた場合、その格子に障害物があると判別する手法である。

$$\max P_i - \min P_i > V_{dif} \quad (1)$$

しかしながら屋外環境においては、走行領域の凹

凸による車体揺れによるノイズ等が生じるため、縁石などの”高さ変化が小さい障害物”的検出が困難となる。そこで、ある注目点まわりの近傍点群に対して主成分分析<sup>(2)</sup>を施し、式2を基に曲率 $\sigma$ を推定することで微小な高さ変化を検出する。

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \quad (2)$$

図2にMin-Max法と曲率推定法を利用し障害物の抽出を行なっている図を示す。図2aがLIDARから得られた情報、図2bが前述した二つの手法より得られた走行不可能な領域(障害物)である。このように障害物を検出することで走行可能な領域の抽出を行う。

#### 意味情報による走行可能領域の検出

芝生や横断歩道などの高さ変化はないが”材質が異なる”ことにより走行可能領域と不可能領域に分けられる環境において、LIDARから得られる情報のみで走行可能領域を検出することは困難である。そこで、我々はステレオカメラから得られた視覚情報により走行環境の認識を行う。画像の分類にはLiang-Chieh Chenらが提唱する手法<sup>(3)</sup>を利用する。今回は、Cityscapes Dataset<sup>(4)</sup>を用いて学習を行い、屋外の走行環境に関連した15クラスの分類を行った。本システムでは、road及びsidewalkと判定された領域を走行可能、terrain及びvegetationと判定された領域を走行不可能領域とする。さらに、3次元空間上での各画素の位置を把握するために、深度画像をもとにセグメントした画像を3次元復元する。画像上の点(u, v)を空間上の点(X, Y, Z)への変換は、以下の式3に基づいて行う。

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \end{bmatrix}^{-1} \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (3)$$

$f$ は焦点距離、 $c_x$ ,  $c_y$ は光学中心、 $R_{ij}$ は回転行列、 $T_i$ は並進ベクトルを示す。本手法の適用例を図3に示す。図3aに対して前述した分類を行った結果が図3cである。さらに、図3bに分類結果に対して走行可能、不可能を基準に二値化を行い、深度画像を用いた3次元復元を行った結果を示す。ここでは、白色の点群が走行可能領域、黒色の点群が走行不可能領域を示している。

#### 形状情報と意味情報の統合

図4に前述した形状情報と意味情報を統合した走

行可能領域検出の結果を示す。図4cと図4dを比較すると、形状情報の乏しい場所でも意味情報の付加を行うことにより走行可能領域の検出が行えていることが分かる。

**3.1.2 交差点検出** 本節では、前節で抽出した走行可能領域から分岐点を検出する手法<sup>(5)</sup>について述べる。まず始めに、仮想LIDARモデルを用いることにより走行可能領域の端点を推定する。仮想LIDARモデルはS. Thrunによって提唱されたビームモデルを使用する<sup>(6)</sup>。このビームモデルは、距離情報を取得している3D-LIDARの位置に設置する。図5aは実際にビームモデルを設置したものである。1フレーム内のフィルタリングされた点は、以下のように定義される。 $P_i = (x_i, y_i, z_i)$ 。ビームモデルはEq.4で表される。ここで、 $Z_k$ はk番目のビームゾーン領域、 $d_k$ は、ビームゾーンk内の点 $P_j$ 間の最短距離を表す。 $x_{ini}$ ,  $y_{ini}$ はビームモデルの発射点を意味する。

$$Z_k = \left\{ \frac{(k-1)\pi}{n} < \tan^{-1}\left(\frac{y - y_{ini}}{x - x_{ini}}\right) \leq \frac{k\pi}{n}, -40 \leq x \leq 40, -40 \leq y \leq 40, k = 1, 2, 3, \dots, n \right\} \quad (4)$$

$$d_k = \{ \min \sqrt{x_j^2 + y_j^2}, P_j \in Z_k, k = 1, 2, \dots, n \}$$

緑色の線は各ビームゾーンの中央にあり、長さはゾーン内の最短の長さを示す。この緑線の端点を走行可能領域の端点とし、走行可能領域の形状を抽出したものが図5bである。抽出した端点に対してRANSAC<sup>(7)</sup>による直線推定を行う。実際にRANSACによる直線推定を行ったものを図5cに示す。本システムでは、分岐形状を認識するために、法線情報をもとに道を構成する直線のペアリングを行い、2直線の軌道及び長さの平均を有する中心線(図5d)を算出する。複数の直線のペアリングが行われその中心線が交差した場合、交差した点を分岐点の中心とし、そこからの各中心線の角度から分岐形状の推定を行う。

**3.2 自己位置推定** 本節では、提案システムにおける自己位置推定法について述べる。本システムではEdge-node mapとのマッチングを行うため、主成分分析によってオドメトリから直線を抽出し、大まかな移動距離と方位から通過したedgeを推定する。また、前述した分岐点検出により推定された分岐形状とEdge-node mapを比較することで、nodeに到達するごとに自己位置の補正を行う。さらに、抽出したオドメトリと推定したedgeとのマッチングを行うことで、姿勢推定をする。

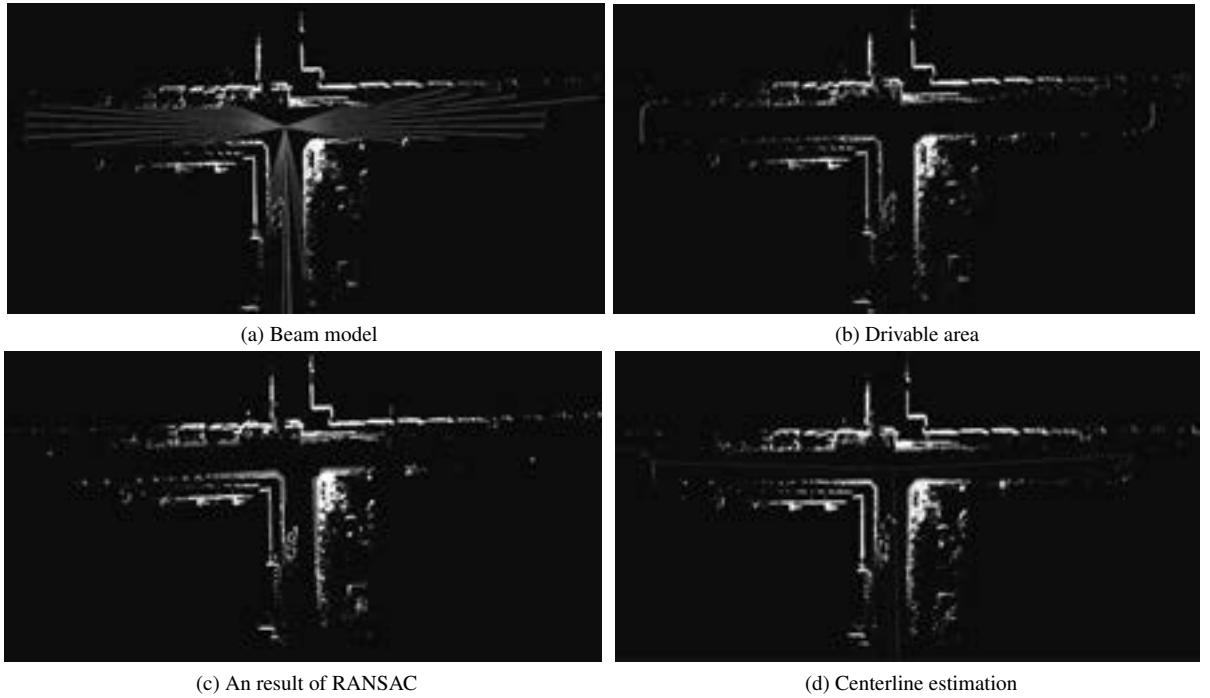


Fig. 5: Intersection detection

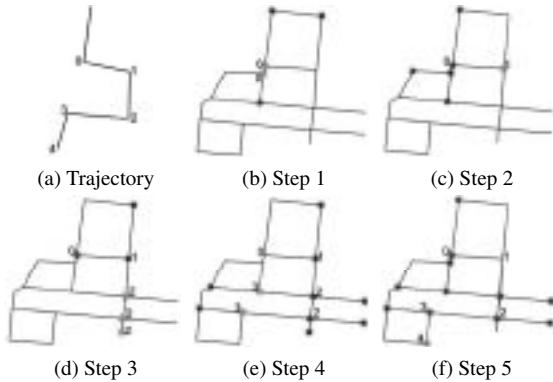


Fig. 6: Process of localization

**3.2.1 Edge-node 推定** 一定数保存したオドメトリ軌跡に対して、主成分分析を行う。分散共分散行列から固有値を算出し、曲率を推定する。ここで曲率は x 軸、y 軸方向の分散を意味する。曲率の値によって、直線的走行と曲線的走行とを分類し、直線を抽出する。抽出した直線に対して、連続性と傾きをもとにクラスタリング処理を行う。そして抽出した直線の長さと進んだ方位をもとに自己位置を推定する。

図 6 に本手法の例を示す。図 6a は得られた軌跡、図 6b～図 6f において黄緑と緑のノードはそれぞれ各ステップで到達可能なノードを示している。ステップ  $i$  番目に推定されたノード  $N_i$  をステップ  $i+1$  番目での初期位置とし、到達確率の高い  $N_{i+1}$  が存在しないとき、 $N_i$  の確率の重みを下げる。これを繰り返すことで、

自己位置の推定を行う。

**3.2.2 姿勢補正** 前述した手法により抽出したオドメトリに対して、連続性と傾きをもとにクラスタリング処理を行う。各直線の交点  $B$  を算出し、これと通過したと推定された node の座標  $N$  を保存する。直線  $B_{i-1}B_i$  と直線  $N_{i-1}N_i$  のなす角を  $\theta$  とし、点  $B$ 、 $N$  へのベクトルを  $\mathbf{B}$ 、 $\mathbf{N}$  とする。また、任意の点を  $x$  平行移動させる行列を  $T(x)$ 、 $B_i$  を中心に  $\theta$  回転させる行列を  $R(\theta)$  とすると、オドメトリの補正值  $C$  は式 5 により求められる。ここで、 $w$  は式 6 により算出されるもので、それぞれ  $w_0$ 、 $r$  は正の定数、 $l$  は線分  $B_{i-1}B_i$  の長さである。

$$C = T(N)R(w\theta)T(w(\mathbf{B} - \mathbf{N})) \quad (5)$$

$$w = \frac{1}{1 + \frac{1-w_0}{w_0} e^{-rl}} \quad (6)$$

**3.3 経路計画** 本節ではロボットが走行する上での指針となる大域経路計画 (Global Path Planning) 及び道なり走行システム、局所的経路計画 (Local Path Planning) について述べる。

**3.3.1 道なり走行** 道なり走行する際には、道の検出だけでなく、検出された道のどの方位に進むべきなのかを知る必要がある。本システムでは作成した Edge-node map から通行する node を選択することで大域経路計画 (Global Path Planning) を行う。この node より、ロボットが進行すべき大まかな方位を決定し、

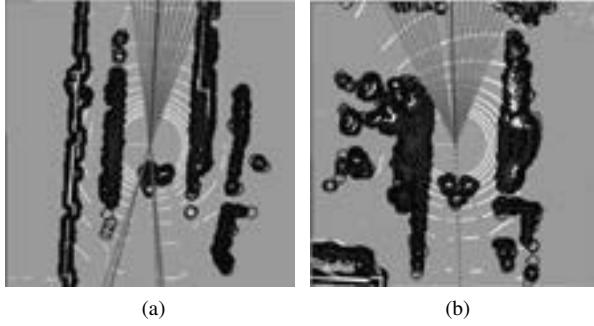


Fig. 7: Local goal decision

交差点が検出されると次の node へ目標方位を更新する。また前節で述べた手法を用いて、図 5 に赤色の線で示すロボットから見た道の続く方位、緑色の線で示すロボットから見たロボットが走行可能な方位を算出する。基本的に道なり走行を行う際には図 7a に示すように進行方向は、算出された道の続く方位に従っている。一方で、車等の障害物や丁字路等への接近により、前方に道の続く方位が算出できなくなった場合やロボットの進行方向から離れた方向に道の方位が算出された場合には、図 7b に赤色の点線で示すようなロボットの走行してきた軌跡から道の方位を推定する。算出された道の方位  $\phi_{ri}$  の中から式 7 より進行方向に最も近い道  $\phi_r$  を選択する。

$$\phi_r = \operatorname{argmin}(|\phi_{ri} - \theta_t|) \quad (7)$$

ここで  $\theta_t$  は node 間を走行しているロボットの走行軌跡である。式 8 より道が続いている方位  $\theta_T$  を推定する。

$$\theta_T = \begin{cases} \phi_r - \theta_{robot} (|\theta_t - \phi_r| < \alpha) \\ \theta_t - \theta_{robot} (|\theta_t - \phi_r| > \alpha) \end{cases} \quad (8)$$

ここで  $\alpha$  は任意の定数である。走行可能領域  $\phi_{ci}$  と推定した  $\theta_T$  を用いて、式 9 ロボットの進行すべき方位  $\phi_{goal}$  を求める。

$$\phi_{goal} = \operatorname{argmin}(|\phi_{ci} - \theta_T|) \quad (9)$$

算出された  $\phi_{goal}$  を図 7 に青色の線として示す。

**3.4 局所的経路計画** 局所的経路計画 (Local Path Planning) ではロボットの運動モデルを考慮した Feruguson や Haward らが提案する手法<sup>(8)(9)(10)</sup>を用いて軌道生成を行う。前節で算出した  $\phi_{goal}$  の方位に目標地点を生成し、軌道上の各点において、式 10 に示す状態パラメータを考慮しロボットが目標状態へと追従するように経路計画を行う。軌道を算出する過程で考慮される運動モデルパラメータは、最大加速度、追従可能な軌道の最大曲率、最大曲率変化である。

$$\mathbf{x}_t = [x_t \ y_t \ \theta_t \ \kappa_t \ v_t] \quad (10)$$

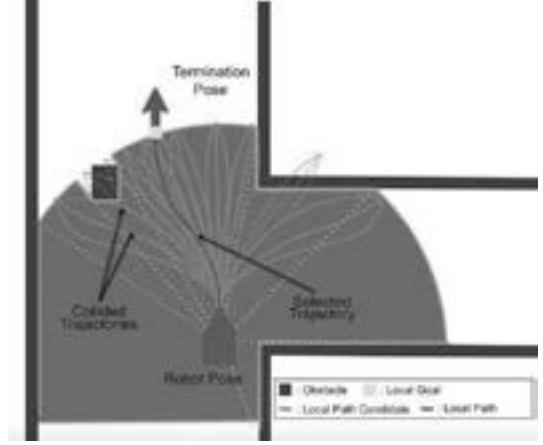


Fig. 8: Local path planning

ここで  $\mathbf{x}_t$  は時刻におけるロボットの状態を表し、 $x_t$ ,  $y_t$ ,  $\theta_t$  はロボットの位置と姿勢、 $\kappa_t$  は軌道曲率、 $v_t$  は前進速度を表している。

運動モデルを考慮して生成される軌道候補群から以下の式 11, 式 12, 式 13 において算出されるコストを最小にする経路を選択する。

$$P = \sqrt{\delta x^2 + \delta y^2} \quad (11)$$

$$H = |\delta \theta| \quad (12)$$

$$Cost = aP + bH \quad (13)$$

ここで、 $P$  及び  $H$  は目標となる終端姿勢との変化量、 $a$ ,  $b$  は任意の定数である。図 8 に概略図を示す。

#### 4. 実験・結果

**4.1 つくばチャレンジ本走行結果** つくばチャレンジ 2018 の本走行において、確認走行区間の突破はできたが、その後の直線の交差点手前で街路樹根元の石に乗り上げてしまい、走行距離 350m という結果になった(図 9)。今回このような結果となった要因としては、道幅を考慮しなかったことにより自己方位の推定が破綻したことが考えられる。3.2 で述べたようにオドメトリから edge の推定を行っているが、道幅の広い箇所では方位がずれていっても直線に走行することができるため、間違った補正が掛かってしまいこのような結果となった。

**4.2 明治大学生田キャンパスにおける実験結果** 提案システムの有用性を示すために明治大学生田キャンパスにおいて自律走行実験を行った。作成した Edge-node map を利用して、ルート上の合計 33ヶ所の交差点と道の検出を行うことで、図 10 に示すルート (1020[m]) を 3 回自律走行させた。自律走行時に推定された自己位置の軌跡を図 11 に示す。自律走行は

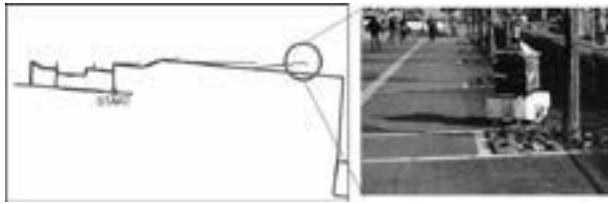


Fig. 9: Result of Tsukuba Challenge2018



Fig. 10: Experimental driving route

3回行ったが、いずれも完走に成功している。また、1020[m]という長距離の自律走行において、自己位置推定の破綻を起こすことなく安定した走行を行うことができた。

## 5. 結 言

本研究では、電子地図から作成した Edge-node map と交差点形状推定を用いることにより、事前形状地図を必要としないナビゲーションシステムを提案した。明治大学生田キャンパスにおける自律走行実験により、LIDAR、ステレオカメラを用いた走行可能領域検出及び交差点形状推定、Edge-node map を利用した自律ナビゲーションシステムの有用性を示した。

## 謝 辞

本研究の一部は、国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の、次世代人工知能・ロボット中核技術開発事業による支援を受けた。ここに御礼申し上げる。

## 参 考 文 献

- (1) S. Thrun, et al. "Stanley: The robot that won the DARPA Grand Challenge." Proc. of the JFR, vol. 23, no. 9, pp. 661-692, 2006.

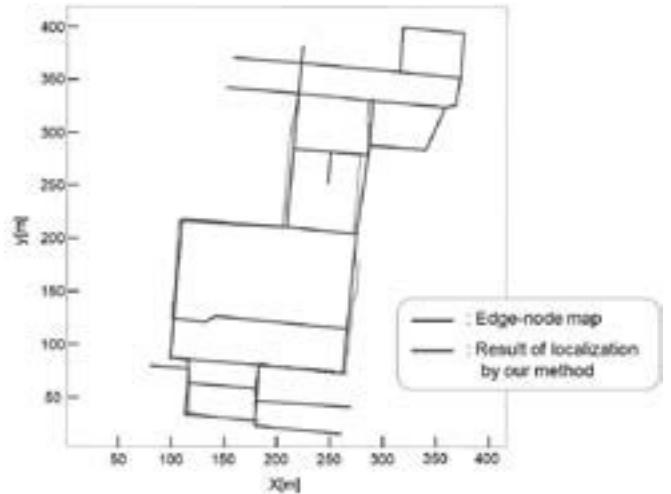


Fig. 11: Result of autonomous driving

- (2) Neuhaus, Frank, et al. "Terrain drivability analysis in 3D laser range data for autonomous robot navigation in unstructured environments." Emerging Technologies and Factory Automation(ETFA), pp. 1-4, 2009.
- (3) Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation." arXiv:1706.05587 (2017).
- (4) Marius Cordts, et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding." arXiv:1604.01685 (2016).
- (5) C. Tongtong, D. Bin, L. Daxue, and L. Zhao, "LiDAR-based long range road intersection detection," in Image and Graphics (ICIG), 2011 Sixth International Conference on.IEEE, 2011, pp.754- 759.
- (6) S. Thrun, and W. Burgard, Probabilistic Robotics, The MIT press, Massachusetts, 2005, pp. 153-153.
- (7) M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," Communications of the ACM, vol. 24, no. 6, pp. 381?395, 1981.
- (8) D.Ferguson, T.M.Howard, and M.Likhachev, "Motionplanning in urban environments," Proc. of the JFR, vol. 25, pp. 939-960, 2008.
- (9) T. M. Howard, A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," Proc. of the IJRR, vol. 26, pp. 141-166, 2007.
- (10) T. M. Howard, C. J. Green, "State space sampling of feasible motions for high- performance mobile robot navigation in complex environments," Proc. of the JFR, vol. 25, pp. 325-345, 2008.

# 建造物の壁に対する相対姿勢を用いた6DoF位置姿勢推定

ロボット工学研究室 指導教員 黒田洋司  
学籍番号 153R153031 4年5組 16番 尾崎亮太

## 1 緒言

移動ロボットのナビゲーションや姿勢制御を行うためには、3次元空間における時々刻々の姿勢を推定する必要がある。移動体の姿勢推定法として、内界センサを使ったデッドレコニングによる推定と、外界センサを使ったSLAMによる推定を統合する手法が提案されている<sup>1)</sup>。このような手法は、初期姿勢に対する相対変化を積算するため、蓄積誤差を補正することが難しいという問題がある。一方、事前環境地図を用意し、その地図とセンサ情報をマッチングすることで蓄積誤差を補正する手法が多く提案されている<sup>2)</sup>。しかしながら、事前地図の使用は、適用可能環境を限定する。

本研究では、従来用いられている慣性センサとSLAMの統合に加え、建造物の鉛直壁面を利用する姿勢推定法を提案する。提案手法は、一般的な建造物の壁がおおよそ鉛直に建てられていることを利用し、事前環境地図を必要とせずに蓄積誤差を補正する。屋外での実機走行実験によって、本研究の有用性を示す。

## 2 提案手法

本手法では、ロボットが静止している状態で慣性センサによって初期姿勢を求め、それに対して慣性センサを用いたデッドレコニングおよびSLAMで推定される相対姿勢変化を積算する。そして、壁面を観測した場合は、ロボット座標系での重力ベクトルを推定し、蓄積誤差を補正する。重力ベクトルの推定及び補正の詳細は2.1~2.3節で記述する。デッドレコニングで推定する姿勢を予測、SLAMで推定する姿勢および壁面に対する相対姿勢を観測とし、拡張カルマンフィルタ(E.K.F.)で統合する。車輪型ロボットで人工環境を走行することを想定しており、各姿勢角度変化が急激なものではなく、E.K.F.で十分扱えると仮定する。

座標系を以下に定義する。

- ロボット座標系：ロボットに固定され、進行方向をx軸の正とする右手直交座標系とする。
- ワールド座標系：ロボット初期姿勢位置において原点がロボット座標系と一致し、重力方向をz軸の負とする右手直交座標系とする。

### 2.1 主法線抽出

LiDARで得られる点群情報に対して主成分分析を用いることで、各注目点に対しての法線ベクトルを算出する。この法線群から、信頼性の高い鉛直面を持つ法線を以下の条件で抽出する。

- 注目点が持つ近傍点の数が十分多い（密度が高い）。
- 1ステップ前の推定重力ベクトル  $\mathbf{G}'_{\text{robot}}$  をもとに、観測された法線  $\mathbf{N}$  が鉛直面を持つかを判定する。

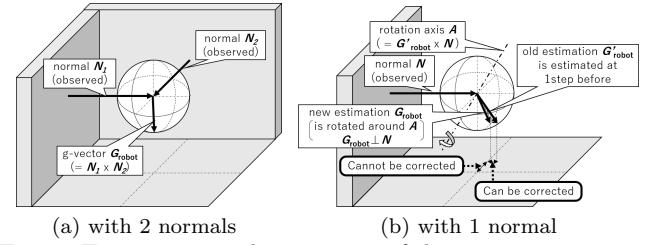


Fig. 1: Estimation and correction of the gravity vector

る（式(1)で算出される角度  $\beta$  が十分小さい）。

$$\beta = \left| \cos^{-1} \frac{\mathbf{N} \cdot \mathbf{G}'_{\text{robot}}}{\|\mathbf{N}\| \|\mathbf{G}'_{\text{robot}}\|} - \frac{\pi}{2} \right| \quad (1)$$

- 法線に垂直な平面と近傍点群の二乗誤差が十分小さい（平面度が高い）。

抽出された法線群をガウス球<sup>3)</sup>に写像し、球内の点群に対してユークリッド距離に基づくクラスタ分析を適用する。本手法では、適当な法線を反転させガウス球を半球のみに反映する。つまり、平行で向かい合う壁面は同じ法線ベクトルを持つようとする。クラスタリング終了後に、メンバが閾値より少ないクラスタは外れ値として除去する。各クラスタが持つ点群の重心を算出し、それらの位置ベクトルを支配的な法線（主法線）として用いる。

### 2.2 重力方向推定

クラスタリングされた主法線を用いてロボット座標系における単位重力ベクトル  $\mathbf{G}_{\text{robot}}$  を推定する。主法線の数に応じて以下のように算出する。

- 主法線の数：3以上  
主法線をガウス球における点として扱い、この点群に対する主成分分析で得た法線を重力ベクトルとする。ただし、各クラスタが持つ点の数で重み付けを行う。
- 主法線の数：2  
2つの法線ベクトル  $\mathbf{N}_1, \mathbf{N}_2$  の外積を重力方向ベクトルとする。図1aに推定の様子を示す。
- 主法線の数：1  
ワールド座標系における絶対的な姿勢を得るには、平行でない鉛直面が2面以上必要であるが、1ステップ前の推定重力ベクトル  $\mathbf{G}'_{\text{robot}}$  を用いて、式(2)のように部分的に推定重力ベクトルを補正し算出する。観測された主法線ベクトルを  $\mathbf{N}$  とする。図1bに推定の様子を示す。

$$\mathbf{G}_{\text{robot}} = \frac{\mathbf{G}'_{\text{robot}} - (\mathbf{G}'_{\text{robot}} \cdot \mathbf{N})\mathbf{N}}{\|\mathbf{G}'_{\text{robot}} - (\mathbf{G}'_{\text{robot}} \cdot \mathbf{N})\mathbf{N}\|} \quad (2)$$

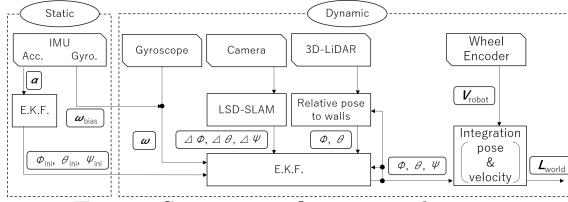


Fig. 2: System configuration diagram

- 主法線の数 : 0

重力方向を推定することが出来ない。

### 2.3 推定姿勢補正

重力ベクトルの推定誤差が、推定姿勢の誤差であることを用いて推定を補正する。補正後の姿勢を表す四元数  $\mathbf{q}_{\text{pose.wall}}$  の算出法を以下で示す。1ステップ前の推定重力ベクトル  $\mathbf{G}'_{\text{robot}}$  と、2.2節で推定した推定重力ベクトル  $\mathbf{G}_{\text{robot}}$ において、この2ベクトル間の回転を表す四元数を  $\mathbf{q}_{\text{error}}$  とする。推定重力ベクトル  $\mathbf{G}'_{\text{robot}}$  に対応する推定姿勢を表す四元数を  $\mathbf{q}'_{\text{pose}}$  としたとき、次のように推定姿勢は補正される。

$$\mathbf{q}_{\text{pose.wall}} = \mathbf{q}'_{\text{pose}} \mathbf{q}_{\text{error}} \quad (3)$$

$\mathbf{q}_{\text{pose.wall}}$  および  $\mathbf{q}'_{\text{pose}}$  はワールド座標系での姿勢であり、 $\mathbf{q}_{\text{error}}$  はロボット座標系での誤差（回転）である。

## 3 実験

起伏のあるコースで移動ロボットを走行させ、提案手法および従来手法により、時々刻々のロボットの自己姿勢を推定する実験を行った。しかし、走行中の全地点での姿勢の真値を得て、推定を評価することは難しいため、式(4)のように、推定姿勢と並進速度を統合することで、走行中の推定姿勢誤差を走行終了時の推定位誤差に置き換えて評価した。 $\mathbf{V}_{\text{robot}}$  はロボット座標系での並進速度ベクトル、 $\mathbf{L}_{\text{world}}$  はワールド座標系でのロボットの推定位ベクトルである。

$$\begin{aligned} \mathbf{L}_{\text{world},k} &= \mathbf{L}_{\text{world},k-1} + \mathbf{Rot}_k \mathbf{V}_{\text{robot},k} dt \\ &= \begin{pmatrix} L_{X_w,k-1} \\ L_{Y_w,k-1} \\ L_{Z_w,k-1} \end{pmatrix} + \mathbf{Rot}_k \begin{pmatrix} V_{X_r,k} \\ 0 \\ 0 \end{pmatrix} dt \end{aligned} \quad (4)$$

本実験での提案手法のシステム図を図2に示す。図中のStaticは、ロボットが静止している状態での初期姿勢推定のシステム、Dynamicは走行開始後のシステムをそれぞれ示している。提案手法におけるE.K.F.の観測で用いるSLAMには、LSD-SLAM<sup>4)</sup>を選定した。LSD-SLAMとは単眼カメラSLAMであり、推定のレートの速さとスケールに依存しないことが姿勢推定に適していると判断し選定した。姿勢推定の従来手法として、慣性センサによるデッドレコニングと、単独のLSD-SLAMによる推定を行った。提案手法およびこれらの従来手法により姿勢をそれぞれ推定し、ホイールエンコーダの並進速度と統合する。本研究の趣旨から外れるため、LSD-SLAMでのループクローズは行わない。ロボットに搭載したセンサは、Velodyne HDL-32E、Xsens MTi30、RealSense D435、ホイールエンコーダである。RealSense D435は、LSD-SLAMのた

Table 1: Errors of position and pose estimations

error in...	proposed method	LSD-SLAM	Gyrodometry
$X_w$ [m]	+1.8843	+4.8471	+2.0761
$Y_w$ [m]	-0.6136	-2.3376	-0.2382
$Z_w$ [m]	+1.9233	+9.0677	-2.6726
Euc.dist.[m]	2.762	10.544	3.393
roll <sub>w</sub> [deg]	-2.069	+6.922	-2.984
pitch <sub>w</sub> [deg]	-0.095	-8.721	+6.970
yaw <sub>w</sub> [deg]	-0.300	+18.257	-0.963

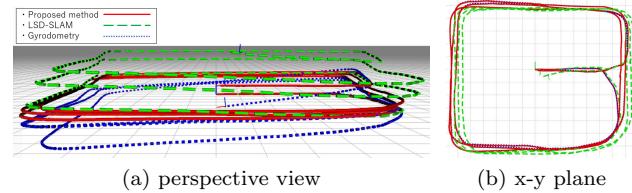


Fig. 3: Estimated trajectories of the robot in the experiment

めに画像のみを利用した。評価に用いる真値には、走行開始時と走行終了時における、モーションキャプチャ(Vicon Vero v1.3X)で得られる位置姿勢を用いる。走行コースは、屋外の1周約250[m]、高低差約3[m]のコースであり、3周走行する。ただしスタート地点とゴール地点はモーションキャプチャを設置した屋内である。

表1に、走行終了時における各手法での推定位姿勢の誤差を示す。表中の誤差は、ワールド座標系の座標軸を基準とした位置および姿勢角の誤差である。また、Euc.dist.は、ユークリッド距離で表した位置推定誤差であり、従来手法に比べて、提案手法の値が小さい。同じ並進速度を用いた場合、姿勢の推定誤差が小さいほど、推定位（軌跡）の誤差が小さくなるため、走行中の姿勢推定の誤差も提案手法の方が小さいと言える。図3に、各手法で推定されたロボットの軌跡を示す。視覚的にも提案手法による推定軌跡の誤差が小さいことがわかる。特に、 $Z_w$ 軸（鉛直）方向の並進誤差が小さく抑えられており、壁面を用いたロール、ピッチ補正による結果だと考えられる。

## 4 結言

従来の慣性センサとSLAMによる移動ロボットの姿勢推定に加え、鉛直に建てられている建造物の鉛直面を利用する姿勢推定法を提案した。また、実験で従来手法と比較することで、提案手法が事前環境地図を用いないで蓄積誤差を補正することが示された。

## 参考文献

- 1) L. von Stumberg, V. Usenko and D. Cremers, "Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization", in *International Conference on Robotics and Automation*, pp.2510–2517, 2018.
- 2) 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, pp.312–328, 2007.
- 3) B.K.P. Horn: "Extended gaussian images", *Proceeding of the IEEE*, vol.72, no.12, pp.1671–1686, 1984.
- 4) J. Engel, J. Stueckler and D. Cremers: "LSD-SLAM: Large-Scale Direct Monocular SLAM", *European Conference on Computer Vision*, pp.834–849, 2014.

# 6-DoF EKF SLAM with global planar features in artificial environments

Ryota Ozaki<sup>\*1</sup> and Yoji Kuroda<sup>\*2†</sup>

This paper presents online SLAM for localization of mobile robots in artificial environments. This proposed method exploits global planar features as landmarks in extended Kalman filter (EKF). The main purpose of using global planar features is reducing accumulative error of the estimation. Planes are extracted from point-cloud of 3-D LiDAR as normals. These normals are projected onto “depth-Gaussian sphere”. Those points from each plane are concentrated in one place on the sphere since planes has many normals which are almost same directions. These concentrations of points are deemed as planar features. The state vector of EKF has states of both a robot and landmarks. Prediction steps compute integration of angular velocity from gyroscope and linear velocity from wheel encoders, respectively. Observation steps update states by using observed planar features. The method associates observed planes and landmark planes, and the state vector is updated. Area of landmark planes are also expanded with every association. Observed planes which are not associated with any landmarks become new landmarks, and the state vector is augmented. Similar landmarks are merged as necessary. To avoid mismatching between observations and landmarks, candidates being associated are selected by some conditions. To evaluate the proposed method, an experiment with an actual robot was performed. It shows the method suppresses accumulative error of localization by using the landmarks.

**Key Words:** Mobile robot, 6-DoF localization, SLAM, EKF, Planar feature

## 1 INTRODUCTION

Estimating the pose of a robot in the surrounding environment is one of the classic problems of mobile robotics. In a known environment, the robot can self-localize by matching sensor information at the moment to the prior information of the environment. Especially, many methods using maps as prior information have been proposed [1]. In those methods, accuracy of the map is important. However, mapping requires accurate estimation of the robot’s pose. This leads a dilemma: for self-localization, the robot requires a map, but to build such a map, the pose of the robot must be known [2]. A solution of this is SLAM (Simultaneous Localization And Mapping) [3]. Many kinds of SLAM have been developed. This paper focuses on online (real-time) SLAM because localizing the pose and correcting estimation online is required when the robot runs in unknown environments. SLAM with scan matching method such like ICP scan matching [4], NDT scan matching [5, 6] is one of well-known online SLAM. Zhang [7] has proposed a matching method based on edge and planar features with good results. Using these features achieves low-drift and low-computational complexity. However it is difficult for scan matching methods to correct accumulative error because it integrates relative pose transformation to the initial pose. On another hand, landmark-based SLAM can suppress error while the robot is observing the landmarks. Taguchi [8] has proposed a method which associates observation and landmarks, and applies SVD to estimate the pose of the robot. But using SVD means noise of observation is not considered. Landmark-based SLAM implemented with EKF is well known [9]. Exploiting planes as landmarks is a well known method in the area of visual SLAM [10, 11, 12]. However they do not describe how to handle landmarks which the robot pass by and can not be observed. It means it is not considered how to deal with the situation that robot comes back to known place, and how to avoid false matching. Some methods use planar landmarks with Manhattan world assumption [13, 14]. The assumption assumes that planes in the environment are orthogonal to each other. This assumption has low versatility

although it linearizes the system and makes estimation easier.

To address these issues above, this paper proposes EKF SLAM with global planar features as landmarks. And data association including the situation that the robot comes back to known landmarks is implemented. Note that the proposed method does not use any prepared maps nor models of the environment. Information of the planar landmarks such like position and orientation are also unknown. And Manhattan world assumption is not used in this method, which means any planes can be used in this method.

Main contributions of this paper are summarized here:

- A method of extracting planes from point-cloud as normals is described.
- EKF framework with planar landmarks is described.
- Data association between sensor observations and landmarks including the revisit situation is described.
- Condition setting for avoiding false matching is described.

## 2 6-DoF EKF SLAM with global planar features in artificial environments

The system configuration diagram of the proposed method is shown in Fig. 1. The method is based on landmark-based SLAM [3]. Prediction step and update step are repeated in EKF. Date association between observations and registered landmarks is needed for the update process. Landmarks means planar features in this method. The planar features are extracted from point-cloud.

### 2.1 Extraction of planes from point-cloud

#### 2.1.1 Generation of normal-cloud

Normal-cloud  $\mathbf{N}$  is generated by applying principal component analysis (PCA)[15] to the local neighbor points of each query point in point-cloud that is obtained with 3-D LiDAR. The neighbor points (query point-cloud)  $\mathbf{C}_{\text{query}}$  are searched by kd-tree[16] with searching radius. This searching radius is set larger at farther point since point density is more sparse in farther area.

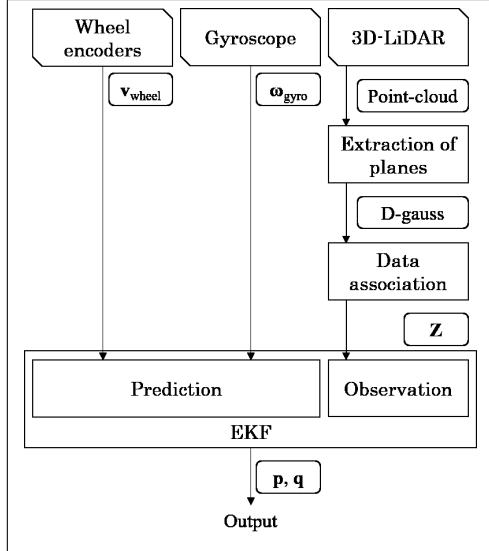
$$\mathbf{C}_{\text{query}} = [\mathbf{c}_0 \ \cdots \ \mathbf{c}_{n_{\mathbf{C}_{\text{query}}}}], \quad \mathbf{c}_i = (c_{i,x} \ c_{i,y} \ c_{i,z}) \quad (1)$$

$$\mathbf{N} = [\mathbf{n}_0 \ \cdots \ \mathbf{n}_{n_{\mathbf{N}}}], \quad \mathbf{n}_i = (n_{i,a} \ n_{i,b} \ n_{i,c} \ n_{i,d}) \quad (2)$$

where  $\mathbf{c}_i$  denotes a point, and  $\mathbf{n}_i$  denotes parameters of the plane (i.e.  $n_{i,a}x + n_{i,b}y + n_{i,c}z + n_{i,d} = 0$ ).

<sup>\*1</sup>Ryota Ozaki is with Graduate School of Science and Technology, Meiji University, Kanagawa, 214-8571, Japan  
ce192021@meiji.ac.jp

<sup>\*2</sup>Yoji Kuroda is with Graduate School of Science and Technology, Meiji University, Kanagawa, 214-8571, Japan  
ykuroda@meiji.ac.jp



**Fig.1:** System architecture

### 2.1.2 Selection from normal-cloud

Query point-clouds which have high flatness are picked up from the normal-cloud with their third eigenvector (normal) by the conditions below.

- The number of query points  $n_{C_{\text{query}}}$  is many enough.

$$n_{C_{\text{query}}} > \text{TH}_{n_{C_{\text{query}}}} \quad (3)$$

- Fitting error between the plane and neighbor points  $e$  is small enough.

$$e = \sum_{i=0}^n \frac{|n_a c_{i,x} + n_b c_{i,y} + n_c c_{i,z} + n_d|}{\|\mathbf{n}\|} < \text{TH}_e \quad (4)$$

### 2.1.3 Generation of depth-Gaussian sphere

All initial points of selected normals  $\hat{\mathbf{N}}$  ( $\in \mathbf{N}$ ) are moved to one origin and generate point-cloud  $C_{\text{d-gauss}}$ . Shimizu [17] calls this point-cloud ‘‘depth-Gaussian sphere’’ (cf. Gaussian sphere [18]).

$$\begin{aligned} C_{\text{d-gauss}} &= [\mathbf{c}_{\text{d-gauss}0} \quad \dots \quad \mathbf{c}_{\text{d-gauss}n_{\hat{\mathbf{N}}}}] \\ \mathbf{c}_{\text{d-gauss}i} &= -n_{i,d} (n_{i,a} \quad n_{i,b} \quad n_{i,c}) \end{aligned} \quad (5)$$

### 2.1.4 Clustering in depth-Gaussian sphere

Euclidean clustering is applied to the point-cloud on depth-Gaussian sphere. And clusters which has enough many numbers of members are extracted as plane. Centroid of each extracted cluster is used as planar feature below.

## 2.2 EKF framework

The state of both the robot and planar landmarks are estimated simultaneously in this EKF. The state vector  $\mathbf{x}$  consists of the state of them as Eq. (6). Prediction process is done with input of wheel encoders and gyroscope. Observation process is done with observation of planar landmarks.

$$\begin{aligned} \mathbf{x} &= (\mathbf{p}^T \quad \mathbf{q}^T \quad \mathbf{m}_0^T \quad \dots \quad \mathbf{m}_n^T)^T \\ \mathbf{p} &= (x_r \quad y_r \quad z_r)^T, \quad \mathbf{q} = (\phi_r \quad \theta_r \quad \psi_r)^T \\ \mathbf{m}_i &= (x_{\text{lm},i} \quad y_{\text{lm},i} \quad z_{\text{lm},i})^T \end{aligned} \quad (6)$$

where  $\mathbf{p}$  denotes the position of the robot,  $\mathbf{q}$  denotes the attitude of the robot, and  $\mathbf{m}_i$  denotes the position of the landmark.

### 2.2.1 Prediction with wheel encoder

Linear velocity measured with wheel encoder  $\mathbf{u}_{\text{wheel}}$  is transformed from the local coordinate to the global coordinate.

$$\mathbf{u}_{\text{wheel}} = \mathbf{v}_{\text{wheel}} = (v_x \quad 0 \quad 0)^T \quad (7)$$

$$f\{\mathbf{x}_k\} = \mathbf{x}_k + \begin{pmatrix} \text{Rot}_{xyz}^{l \rightarrow g}\{\mathbf{q}_k\} \mathbf{u}_{\text{wheel},k} \Delta t \\ \mathbf{0}_3 \\ \mathbf{0}_{3 \times n} \end{pmatrix} \quad (8)$$

where  $\mathbf{v}_{\text{wheel}}$  denotes velocity measured with the wheel encoders, and  $\text{Rot}_{xyz}^{l \rightarrow g}$  is the rotation matrix which transforms the point from the local frame to the global one.

### 2.2.2 Prediction with gyroscope

Angular velocity measured with wheel encoder  $\omega_{\text{gyro}}$  is transformed from the local coordinate to the global coordinate.

$$\mathbf{u}_{\text{gyro}} = \omega_{\text{gyro}} = (\omega_x \quad \omega_y \quad \omega_z)^T \quad (9)$$

$$f\{\mathbf{x}_k\} = \mathbf{x}_k + \begin{pmatrix} \mathbf{0}_3 \\ \text{Rot}_{\phi\theta\psi}^{l \rightarrow g}\{\mathbf{q}_k\} \mathbf{u}_{\text{gyro},k} \Delta t \\ \mathbf{0}_{3 \times n} \end{pmatrix} \quad (10)$$

where  $\omega_{\text{gyro}}$  denotes angular velocity measured with the gyroscope, and  $\text{Rot}_{\phi\theta\psi}^{l \rightarrow g}$  is the rotation matrix which transforms the angles from the local frame to the global one.

### 2.2.3 Observation of planar landmarks

Observation process is done when observations of planar features are associated with known landmarks. When the observations is not associated with any landmarks, the state vector is augmented and the state of the new landmark is added. How to associate between observations and landmarks is described at next section.

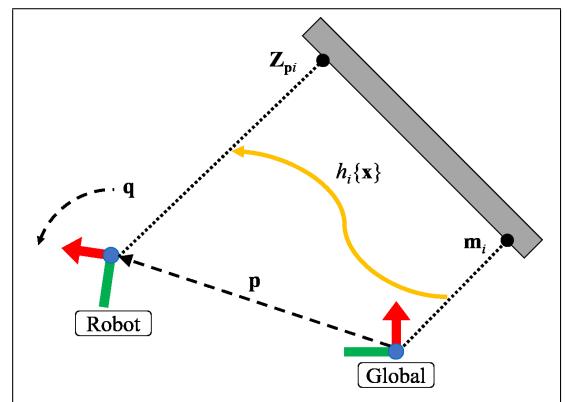
$$\mathbf{z} = (\mathbf{z}_0^T \quad \dots \quad \mathbf{z}_n^T)^T \quad (11)$$

$$\mathbf{z}_i = -n_{i,d} (n_{i,a} \quad n_{i,b} \quad n_{i,c})^T$$

$$\mathbf{z}_p = (\mathbf{z}_{p_0}^T \quad \dots \quad \mathbf{z}_{p_n}^T)^T$$

$$\mathbf{z}_{p_i} = h_i\{\mathbf{x}_k\} = \text{Rot}_{xyz}^{g \rightarrow l}\{\mathbf{q}_k\}(\mathbf{m}_{i,k} - \frac{\mathbf{p}_k \cdot \mathbf{m}_{i,k}}{\|\mathbf{m}_{i,k}\|^2} \mathbf{m}_{i,k}) \quad (12)$$

where  $\mathbf{z}$  is the observation vector, and  $\mathbf{z}_p$  is the predicted observation vector. Fig. 2 shows the function of  $h_i\{\mathbf{x}\}$ .



**Fig.2:** Predicted observation

## 2.3 Data association

### 2.3.1 Search for a corresponding landmark

Each registered landmark searches observation which has minimum Mahalanobis distance  $d_{\text{Mahalanobis,min}}$ . When the distance is smaller than the threshold, the observation is associated with the known landmark. When it is larger, the observation is registered as a new landmark.

$$\begin{aligned} d_{\text{Mahalanobis,min}} &= \mathbf{y}_i^T \mathbf{S}_i^{-1} \mathbf{y}_i < \text{TH}_{d_{\text{Mahalanobis}}} \\ \mathbf{y} &= \mathbf{z} - \mathbf{h}, \quad \mathbf{S} = \mathbf{J}_h \mathbf{P} \mathbf{J}_h^T + \mathbf{R} \end{aligned} \quad (13)$$

where  $\mathbf{J}_h$  is the Jacobian of the vector  $\mathbf{h}$ ,  $\mathbf{P}$  is the covariance matrix, and  $\mathbf{R}$  is the noise matrix of observation.

### 2.3.2 Registration of a new landmark

A new landmark is registered and initialized with the information below:

- Own coordinate (origin)  
The point which the landmark is observed at the first time becomes its origin of the own coordinate.
- Observed range  
Each landmark records the range which the robot has observed it before in the coordinate of the landmark.
- Direction of normal  
Planes can have two direction of normal. But only one side of the planes can be observed in the real world because walls and other things have thickness. Therefore the direction of the plane is registered when the landmark is initialized.

### 2.3.3 Update of information of associated landmark

When observation is associated with the observation, the information of the landmark below is updated.

- Orientation of coordinate
- Observed range

### 2.3.4 Merge of landmarks

Landmarks are merged when two or more landmarks find a same observation as a corresponding one. The oldest landmark which is observed earliest is remained, and the others are absorbed to the oldest one. The range which the robot has observed the landmark is also merged. This merge can happen when the robot comes back to the known place.

However, landmarks are not merged when these landmarks have been observed at the same time at prior step. At that time, only the landmark whose Mahalanobis distance  $d_{\text{Mahalanobis},\min}$  is smaller is associated, and the others have no pair.

### 2.3.5 Narrowing down the candidates

In order to avoid false association, registered landmarks are narrowed down every step before association process above. Geometric constraints that the method exploits are here:

- Only one side of the plane is visible and it is not visible from the other side.
- The landmark is not visible from a far position from the last position of the robot which the landmark is observed (observed range) like Fig. 3.

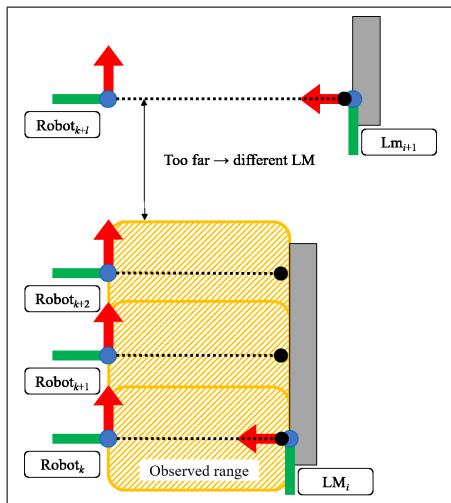


Fig.3: Observed range

## 3 EXPERIMENTS

### 3.1 Experimental outline

In this experiment, the 6-DoF pose (position and attitude) of the robot was estimated by the proposed method and comparative approaches while the robot moved. It is hard to get all ground truth of the pose while the robot is moving. Hence, the robot was driven back to the exact starting position. Thus, the return position error and the return attitude error were evaluated instead of the whole trajectory.

### 3.2 Experimental conditions

#### 3.2.1 Hardware

The experimental mobile robot is shown in Fig. 4. It has Velodyne HDL-32ECXsens MTi30 and wheel encoders.

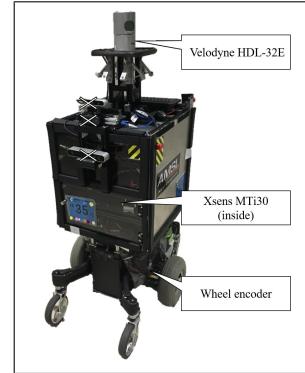


Fig.4: The experimental robot

#### 3.2.2 Environment

The experimental floor map is shown in Fig. 5. The robot was driven for three rounds of this course.

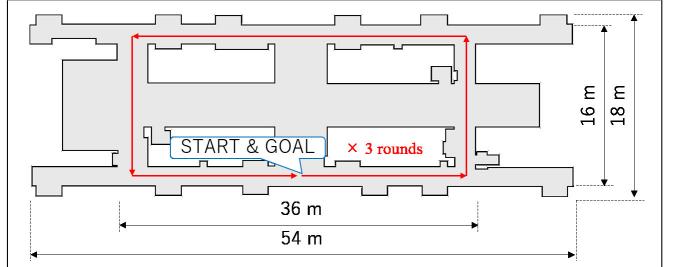


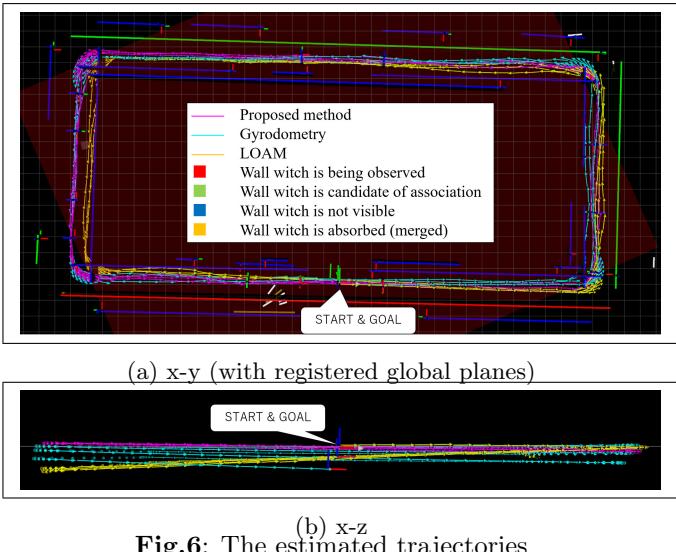
Fig.5: The experimental floor map

### 3.3 Experimental result

Trajectories estimated by the proposed method and the comparative approaches on x-y plane are shown in Fig. 6a. The trajectories on x-z plane are shown in Fig. 6b. The return position error and the return attitude error are shown in Table 1. The proposed method estimated the pose of the robot with less error compared with the others. It was stable by using global planar features. The method could correct the estimation as long as the robot observes the registered planar landmarks. And some landmarks were merged when the robot came back to the start point from the other side. On the other hand, it was hard for gyrodometry and LOAM to correct accumulative error while it was driven. Although LOAM estimated transformation of the pose well each step, once it detected bad matching, it was hard to correct, which comes from the feature of scan matching method.

## 4 CONCLUSIONS AND FUTURE WORK

6-DoF EKF SLAM with global planar features in artificial environments was proposed. By measuring the position of planar landmarks, 6-Dof robot pose and the position of associated



(a) x-y (with registered global planes)

(b) x-z  
Fig.6: The estimated trajectories

**Table 1:** Return position error and the return attitude error

error in...	Proposed method	Gyrodometry	LOAM
$x[m]$	+0.005	-0.632	-0.117
$y[m]$	+0.041	+0.036	+0.077
$z[m]$	-0.088	-1.212	-0.574
$d_{Euc}[m]$	0.097	1.367	0.590
$\phi[\deg]$	-0.6	-2.1	-1.3
$\theta[\deg]$	+0.3	+1.2	-2.7
$\psi[\deg]$	-1.2	-2.7	-1.4

global planes are updated. The experiment showed that the proposed method had less accumulative error of estimation than comparative approaches.

Since bundle adjustment is not implemented although the proposed method merges landmarks when the revisits known places, the future work of this paper is developing a method to fix motion estimation drift by closing the loop. Another future work is applying the proposed method to outdoor environments. Outdoor ground is not complete plane. Therefore the method often does false matching so far.

## 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] 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, pp.312–328, 2007.
- [2] P. Skrzypczynski, Simultaneous localization and mapping: A feature-based probabilistic approach, *International Journal of Applied Mathematics and Computer Science*, Vol.19, No.4, pp.575–588, 2009.
- [3] S. Thrun, W. Burgard and D. Fox, *probabilistic robotics*, The MIT Press, pp.309–336, 2005.
- [4] S. Rusinkiewicz and M. Levoy, Efficient Variants of the ICP Algorithm, *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pp.145–152, 2001.
- [5] P. Biber and W. Straßer, The Normal Distributions Transform: A New Approach to Laser Scan Matching, *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol.3, pp.2743–2748, 2003.
- [6] M. Magnusson, A. Lilienthal and T. Duckett, Scan registration for autonomous mining vehicles using 3D]NDT, *Journal of Field Robotics*, Vol.24, No.10, pp.803–827, 2007.
- [7] J. Zhang and S. Singh, LOAM: Lidar Odometry and Mapping in Real-time, in *Robotics: Science and Systems Conference*, pp.161–195, 2014.
- [8] Y. Taguchi, Y.D. Jian, S. Ramalingam and C. Feng, Point-Plane SLAM for Hand-Held 3D Sensors, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pp.5182–5189, 2013.
- [9] S. Huang and G. Dissanayake, Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM, *IEEE Transactions on Robotics*, Vol.23, No.5, pp.1036–1049, 2007.
- [10] J. Martinez-Carranza and A. Calway, Unifying Planar and Point Mapping in Monocular SLAM, *Proceedings of the British Machine Vision Conference (BMVC)*, pp.43.1–43.11, 2010.
- [11] A. P. Gee, D. Chekhlov, WW. Mayol-Cuevas, A. Calway, Discovering Planes and Collapsing the State Space in Visual SLAM, *Proceedings of British Machine Vision Conference*, pp.1–10, 2007.
- [12] A. P. Gee, D. Chekhlov, A. Calway, W. Mayol-Cuevas Discovering Higher Level Structure in Visual SLAM, *IEEE Transactions on Robotics*, Vol.24, No.5, pp.980–990, 2008.
- [13] P. Kim, B. Coltin and H. J. Kim, Linear RGB-D SLAM for Planar Environments, *Proceedings of The European Conference on Computer Vision (ECCV)*, pp.333–348, 2018.
- [14] K. Takeuchi, M. Hashimoto and K. Takahashi, Laser-Based-3D Mapping by Human Walking in an Indoor Environment, *The Science And Engineering Review Of Doshisha University*, Vol.53, No.2, pp.84–91, 2012 (in Japanese).
- [15] M. Pauly, M. Gross and L.P. Kobbelt, Efficient simplification of point-sampled surfaces, *Proceedings of the conference on Visualization*, pp.163–170, 2002.
- [16] J.L. Bentley, Multidimensional binary search trees used for associative searching, *Communications of the ACM*, Vol.18, No.9, pp.509–517, 1975.
- [17] S. Shimizu and Y. Kuroda, High-speed registration of point clouds by using dominant planes, *Proceedings of the 19th Robotics Symposia* Cpp.453–458, 2014 (in Japanese).
- [18] B.K.P. Horn, Extended gaussian images, *Proceeding of the IEEE*, Vol.72, No.12, pp.1671–1686, 1984.
- [19] J. Borenstein and L. Feng, Gyrodometry: a new method for combining data from gyros and odometry in mobile robots, *Proceedings of IEEE International Conference on Robotics and Automation*, pp.423–428, 1996.