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

図 目 次

表 目 次

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，大きな分散が連続している間は，推定値が棄却され続けるため，姿勢が推定されない．この先行研究では，静止画像に対する静的推定にのみ着目したため，その点は問題にはならなかった．しかし，リアルタイム推定では，推定が出力されない時間は問題となる．もう一つの問題は，この先行研究ではシミュレーションデータのみを用いた評価しか行われていないことである．

この DNN を実データでのリアルタイム推定に適用するために，本論文では，実データでファインチューニング（fine-tuning）された DNN と，角速度センサを，EKF（extended Kalman filter）[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．

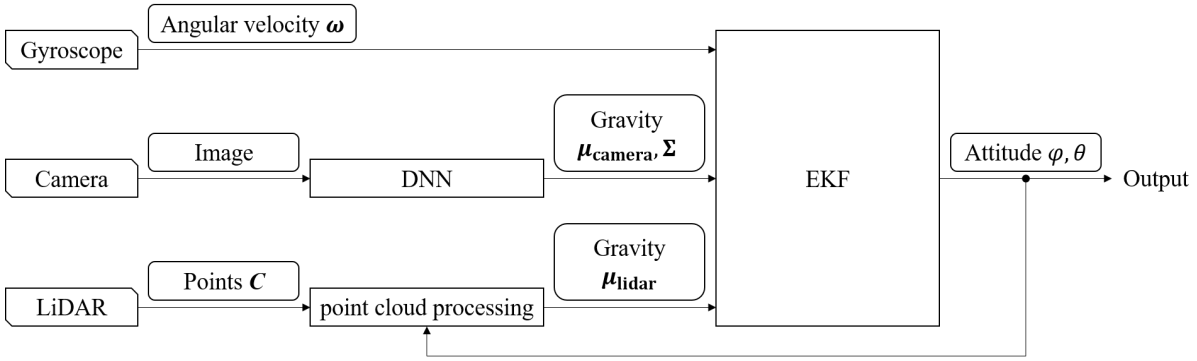


図 2.1: Proposed EKF architecture.
Gyroscopic angular rates are integrated in the prediction process in the EKF. The gravity vector estimated with a LiDAR is integrated in the update process in the EKF. The DNN outputs are integrated in another update process in the EKF.

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．

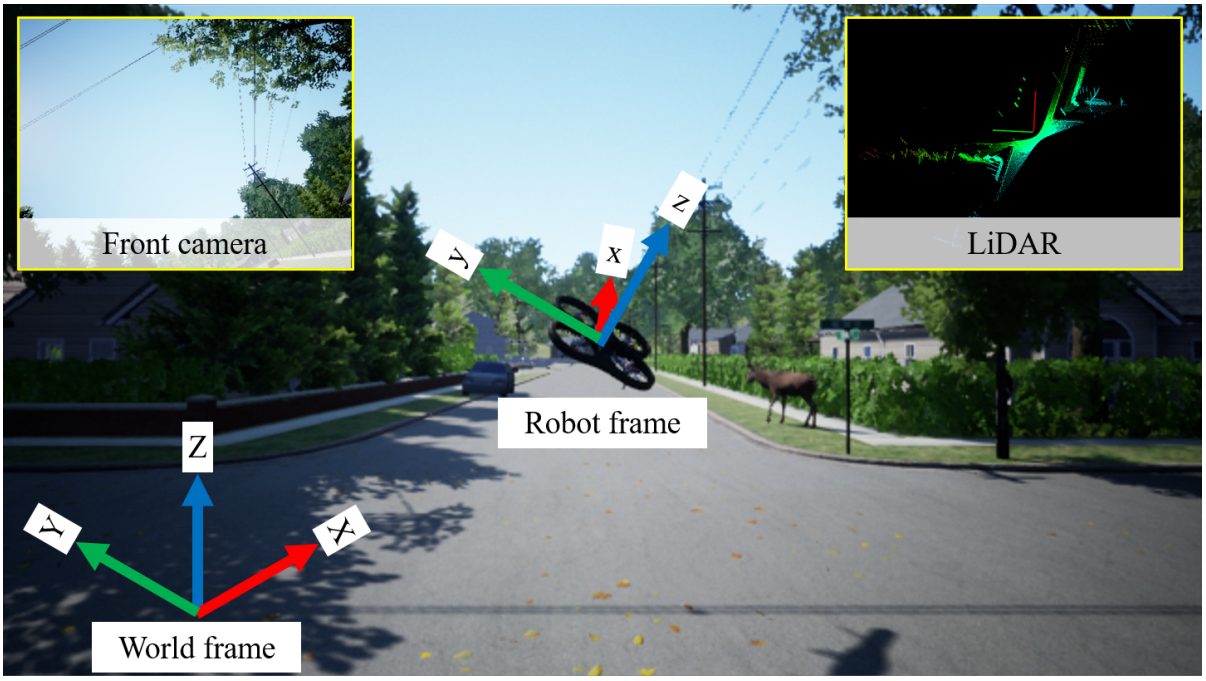


図 2.2: Coordinate description.
This study defines a robot frame and a world frame. The object of the study is estimating the attitude of the robot frame in real-time.

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 ϕ とピッチ角 θ の範囲は，それぞれ [-30 deg, 30 deg] に制限される．

実データセットは，IMU(Xsens MTi-30) とカメラ (RealSense D435) を搭載したセンサースイートを用いて収集する（図 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．

(a) Synthetic data

(b) Real data

図 2.3: Examples of datasets.
The dataset consists of images and corresponded gravity vectors $\boldsymbol{g}[\mathrm{m/s^2}]$ in the robot frame. The examples in (a) were collected in 'Neighborhood' of AirSim. The robot pose and weather parameters are randomized for creating the dataset. The examples in (b) were collected in the campus of Meiji University.

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．

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

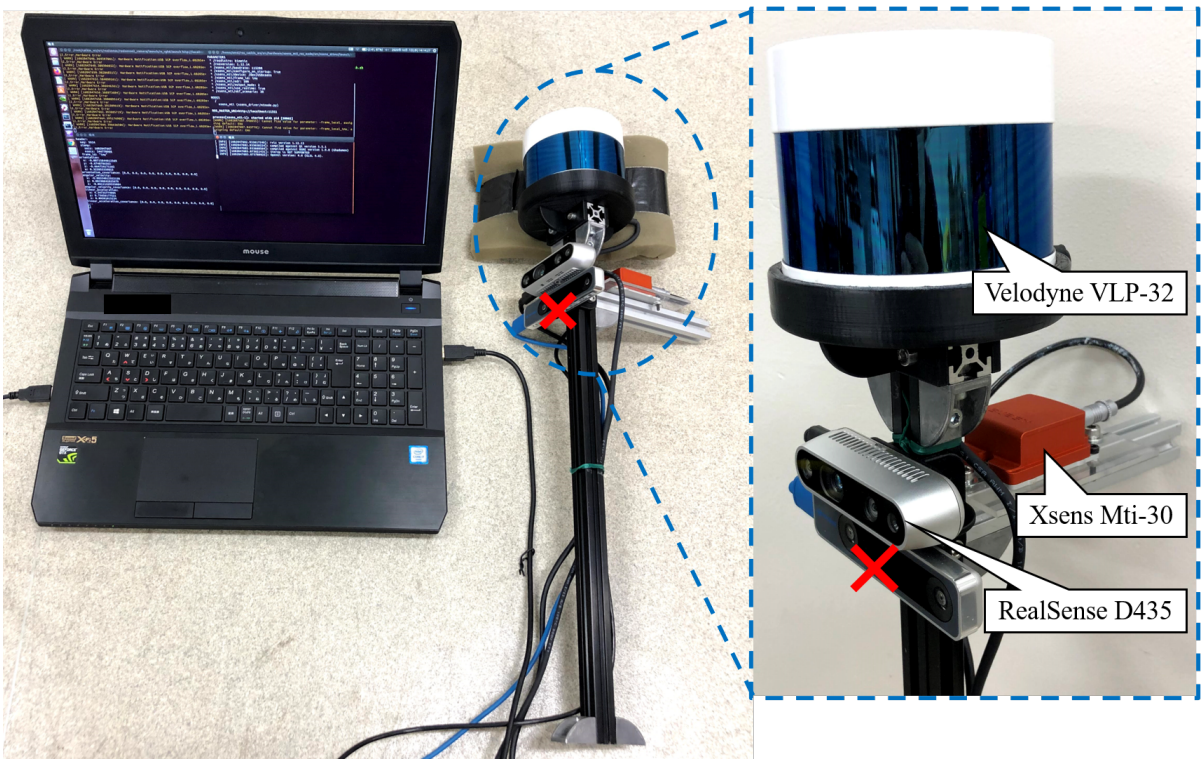


図 2.4: Sensor suite.
Images and acceleration are recorded with this sensor suite when it is still. The judge whether it is still is processed by programing. Note that depth information from RealSense D435 is not used in this study although it is a RGB-D camera.

ここで，h，w はそれぞれ元画像の高さと幅，$\mathrm{FOV}^{\mathrm{v}}$ $(< 2\pi)$ はカメラの鉛直視野角（field-of-view）を表す．さらに，ロールデータを水増しするために，画像をランダムに回転させる．回転角度 $\Delta\phi$ は，[-10 deg, 10 deg] に制限される．それらの処理の後，画像は 224×224 にリサイズされる．RGB 値は，$\boldsymbol{mean} = (0.5, 0.5, 0.5)$，$\boldsymbol{std} = (0.5, 0.5, 0.5)$ に従うように正規化される．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．

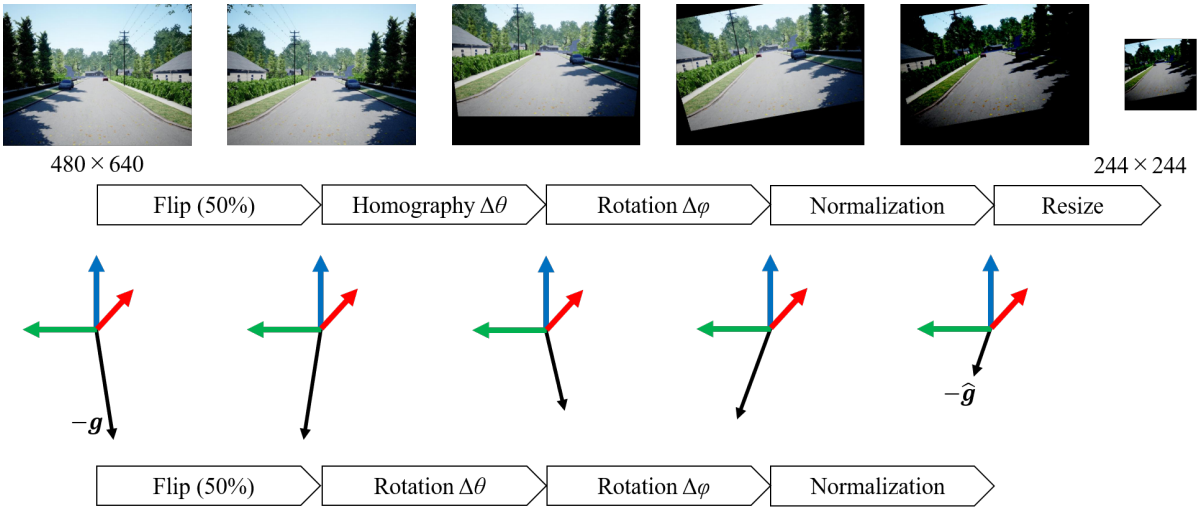


図 2.5: Data preprocessing.
Each input image is randomly flipped, transformed and rotated according to $\Delta\theta$ and $\Delta\phi$. This example shows an image when $\Delta\theta = \Delta\phi = 10$ deg. It is also resized, and is normalized. Each label vector is also transformed acording to the image 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，学習を効率化するめに L2 正規化も適用される．

$$\hat{\boldsymbol{g}} = \begin{cases} \boldsymbol{Rot}^{\mathrm{x}}_{(-\Delta\phi)} \boldsymbol{Rot}^{\mathrm{y}}_{(-\Delta\theta)} \frac{\boldsymbol{g}}{|\boldsymbol{g}|} & (\mathrm{w/o\ flip}) \\ \boldsymbol{Rot}^{\mathrm{x}}_{(-\Delta\phi)} \boldsymbol{Rot}^{\mathrm{y}}_{(-\Delta\theta)} \frac{(g_x, -g_y, g_z)^{\mathrm{T}}}{|\boldsymbol{g}|} & (\mathrm{w/\ flip}) \end{cases}$$
$$\boldsymbol{Rot}^{\mathrm{x}}_{(\Delta\phi)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\Delta\phi) & -\sin(\Delta\phi) \\ 0 & \sin(\Delta\phi) & \cos(\Delta\phi) \end{pmatrix}, \quad \boldsymbol{Rot}^{\mathrm{y}}_{(\Delta\theta)} = \begin{pmatrix} \cos(\Delta\theta) & 0 & \sin(\Delta\theta) \\ 0 & 1 & 0 \\ -\sin(\Delta\theta) & 0 & \cos(\Delta\theta) \end{pmatrix} \tag{2.2}$$

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

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 $(\mu_x, \mu_y, \mu_z, L_0, \cdots, L_5)$ であり，平均ベクトル $\hat{\boldsymbol{\mu}}$ と共分散行列 $\boldsymbol{\Sigma}$ は，それぞれ式 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 $\boldsymbol{L}$ は，対角成分に正の値を持つ必要があるため，それらに指数関数を適用する．

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

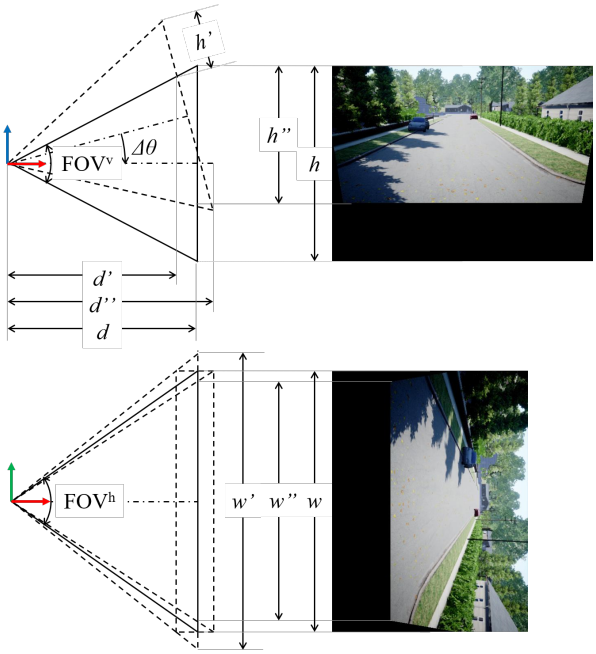


図 2.6: Homography.
The pitch data of the dataset is augmented by the homograpgy transformation.

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

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 $\hat{\boldsymbol{\mu}}_\iota$，共分散行列 $\boldsymbol{\Sigma}_\iota$ が与えられたとき，ラベルデータ $\hat{\boldsymbol{g}}_\iota$ に対する確率密度は以下のように計算される．

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

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，$p(\hat{\boldsymbol{g}}_\iota|\hat{\boldsymbol{\mu}}_\iota, \boldsymbol{\Sigma}_\iota)$ を最大化するように訓練する．データセット $D^{\mathrm{label}} =$

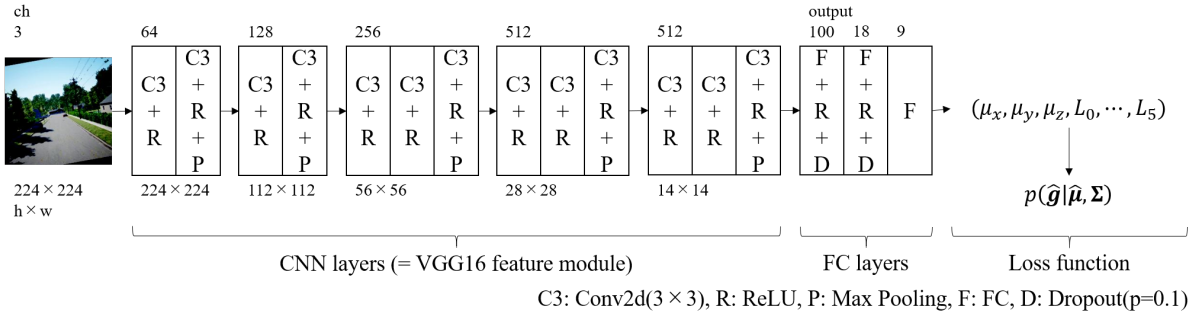


図 2.7: Proposed network architecture.
It consists of CNN layers and FC layers. The input data is a resized image, and the output data are a mean vector and a covariance matrix. They are computed with an output from the final layer as Eq.2.3, 2.4, respectively. Log-probability of multivariate normal distribution is used as a loss function of this model.

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

$$p_{\text{total}} = \prod_{\iota=0}^{\#D} p(\hat{\boldsymbol{g}}_\iota | \hat{\boldsymbol{\mu}}_\iota, \boldsymbol{\Sigma}_\iota) \tag{2.6}$$

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．

$$p_{\text{total}}^{\log} = \sum_{\iota=0}^{\#D} \ln p(\hat{\boldsymbol{g}}_\iota | \hat{\boldsymbol{\mu}}_\iota, \boldsymbol{\Sigma}_\iota) \tag{2.7}$$

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 $l_{(\boldsymbol{\Theta})}$ は以下のように定義される．

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

ここで，$\boldsymbol{\Theta}$ 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 $\boldsymbol{\Theta}$ を更新する．

2.2.5 最適化

パラメータ $\boldsymbol{\Theta}$ の最適化には，Adam（adaptative moment estimation）[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_{CNN} は CNN 層に適用される学習率，lr_{FC} は FC 層に適用される学習率である．実データを用いたファインチューニングでは，学習率はより小さく設定され，それぞれ $lr_{\text{CNN}} = 1 \times 10^{-6}$，$lr_{\text{FC}} = 1 \times 10^{-5}$ である．

2.2.6　不確かさの表現

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

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

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 $\boldsymbol{C}$ とする．

$$\boldsymbol{C} = \{\boldsymbol{c}_0, \cdots, \boldsymbol{c}_i, \cdots, \boldsymbol{c}_{\#\boldsymbol{C}}\}, \quad \boldsymbol{c}_i = \begin{pmatrix} c_{i,x} & c_{i,y} & c_{i,z} \end{pmatrix} \tag{2.10}$$

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

$$r^i = \alpha |\boldsymbol{c}_i| \tag{2.11}$$

$$\boldsymbol{B}^i = \left\{\cdots, \boldsymbol{c}_k, \cdots, \boldsymbol{c}_{\#\boldsymbol{B}^i}\right\} \tag{2.12}$$

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

$$\boldsymbol{N} = \{\boldsymbol{n}_0, \cdots, \boldsymbol{n}_i, \cdots, \boldsymbol{n}_{\#\boldsymbol{C}}\}, \quad \boldsymbol{n}_i = \begin{pmatrix} n_{i,a} & n_{i,b} & n_{i,c} & n_{i,d} \end{pmatrix} \tag{2.13}$$

ここで，$\boldsymbol{n}_i$ 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 $\check{\boldsymbol{N}}$（$\in \boldsymbol{N}$）とする．

$$\check{\boldsymbol{N}} = \left\{ \cdots, \boldsymbol{n}_{\check{i}}, \cdots, \boldsymbol{n}_{\#\check{\boldsymbol{N}}} \right\} \tag{2.14}$$

- 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．

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

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

- 近傍点の数が十分多い．

$$\#\boldsymbol{B}^{\check{i}} > \mathrm{TH}_{\#\boldsymbol{B}} \tag{2.16}$$

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

- 法線 $\boldsymbol{n}_{\check{i}}$ 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．

$$\beta^{\check{i}} = \left| \cos^{-1} \frac{\boldsymbol{n}_{\check{i}} \cdot \boldsymbol{\mu}_{t-1}}{|\boldsymbol{n}_{\check{i}}||\boldsymbol{\mu}_{t-1}|} - \frac{\pi}{2} \right| < \mathrm{TH}_\beta \tag{2.17}$$

  ここで，$\beta^{\check{i}}$ は法線 $\boldsymbol{n}_{\check{i}}$ 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，TH_β 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 $\check{\boldsymbol{N}}$ の全ての始点を，原点に移動させて点群 $\boldsymbol{S}$ 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．

$$\boldsymbol{S} = \left\{ \cdots, \boldsymbol{s}_{\check{i}}, \cdots, \boldsymbol{s}_{\#\check{\boldsymbol{N}}} \right\}, \quad \boldsymbol{s}_{\check{i}} = -n_{\check{i},d} \begin{pmatrix} n_{\check{i},a} & n_{\check{i},b} & n_{\check{i},c} \end{pmatrix} \tag{2.18}$$

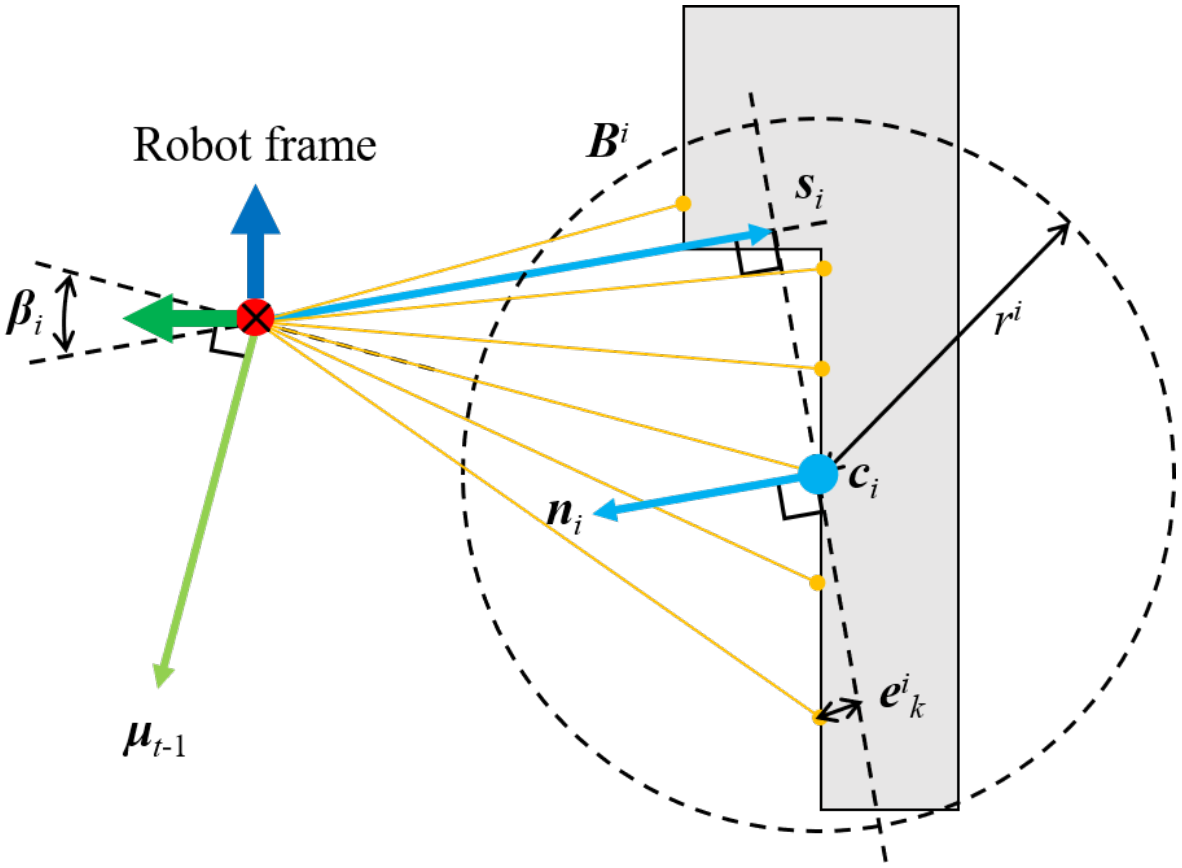


図 2.8: Depth-Gaussian sphere.

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

2.3.3 depth-Gaussian sphere のクラスタリング

点群（depth-Gaussian sphere）$\boldsymbol{S}$ に対して，角度ベースのクラスタリングを適用する（cf. 我々の先行研究 [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，$\boldsymbol{W}$ はクラスタの集合，$\boldsymbol{w}_j$ は環境内の各平面（壁）を表す．

$$\boldsymbol{W} = \{\boldsymbol{w}_0, \cdots, \boldsymbol{w}_j, \cdots, \boldsymbol{w}_{\#\boldsymbol{W}}\}, \quad \boldsymbol{w}_j = \left\{\boldsymbol{s}^j_0, \cdots, \boldsymbol{s}^j_l, \cdots, \boldsymbol{s}^j_{\#\boldsymbol{w}_j}\right\} \tag{2.19}$$

式 2.20 を満たすとき，式 2.21 のように点 $\boldsymbol{s}_{\tilde{i}}$ はクラスタ $\boldsymbol{w}_j$ に分類される．

$$\begin{aligned} &\gamma^j = \min\{\gamma^j_+, \gamma^j_-\} < \mathrm{TH}_\gamma \\ &\gamma^j_+ = \cos^{-1}\frac{\boldsymbol{s}_{\tilde{i}} \cdot \sum_{l=0}^{\#\boldsymbol{w}_j} \boldsymbol{s}^j_l}{|\boldsymbol{s}_{\tilde{i}}||\sum_{l=0}^{\#\boldsymbol{w}_j} \boldsymbol{s}^j_l|}, \quad \gamma^j_- = \cos^{-1}\frac{-\boldsymbol{s}_{\tilde{i}} \cdot \sum_{l=0}^{\#\boldsymbol{w}_j} \boldsymbol{s}^j_l}{|-\boldsymbol{s}_{\tilde{i}}||\sum_{l=0}^{\#\boldsymbol{w}_j} \boldsymbol{s}^j_l|} \end{aligned} \tag{2.20}$$

$$\boldsymbol{w}_j += \boldsymbol{s}_{\tilde{i}} \tag{2.21}$$

式 2.20 を満たさないときは，式 2.22 のように点 $\boldsymbol{s}_{\tilde{i}}$ は新しいクラスタとして登録される．

$$\boldsymbol{W} += \{\boldsymbol{s}_{\tilde{i}}\} \tag{2.22}$$

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 γ^j の例を図 1 に示す．γ^j_+ だけでなく γ^j_- も計算する理由は，次項でクラスターの外積を計算するときに，同一直線上のベクトルの外積を避けるためである．

上記のクラスタリング後，メンバ数が少ないクラスタは無視されるため，次項では，式 2.23 で定義されるクラスター $\check{\boldsymbol{W}}$ が重力ベクトルの推定に使用されることになる．

$$\check{\boldsymbol{W}} = \left\{ \cdots, \boldsymbol{w}_{\check{j}}, \cdots, \boldsymbol{w}_{\#\check{\boldsymbol{W}}} \right\}, \quad \#\boldsymbol{w}_{\check{j}} > \mathrm{TH}_{\#\boldsymbol{w}} \tag{2.23}$$

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

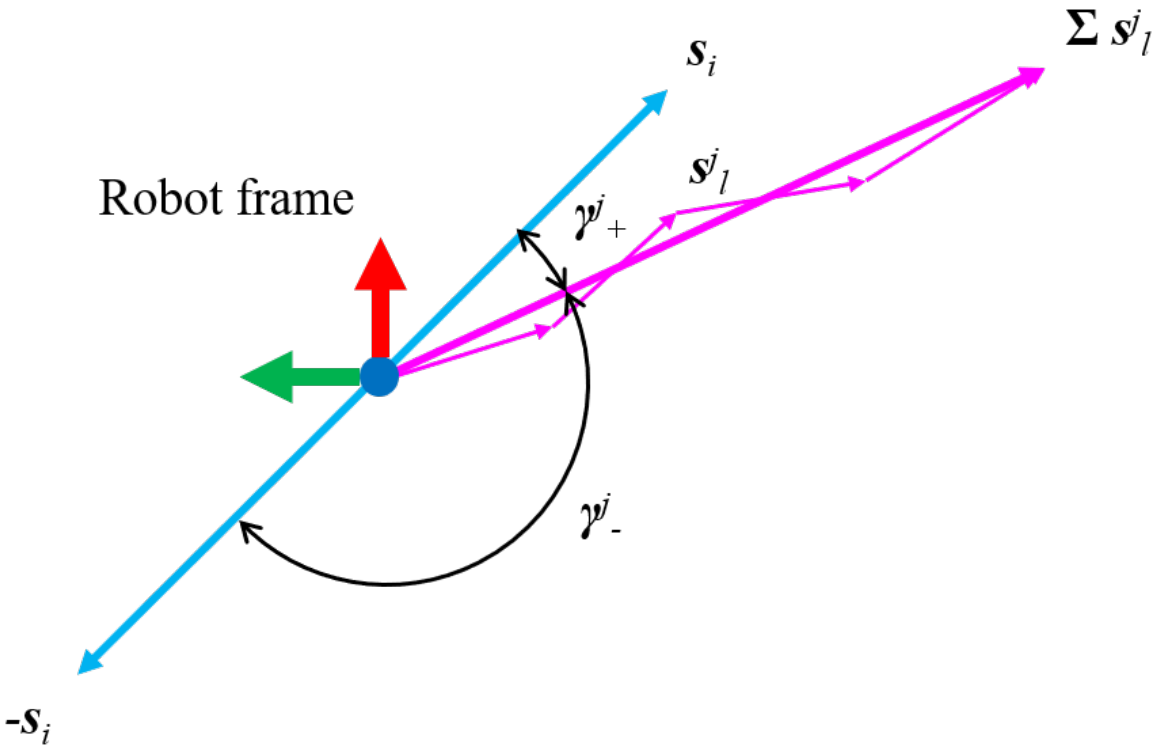


図 2.9: Angle-based clustering.
Points of depth-Gaussian sphere are clustered based on the angle γ^j_+ and γ^j_-. Points with larger norm affect the direction of each cluster more.

2.3.4 鉛直面を用いた重力方向推定

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

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

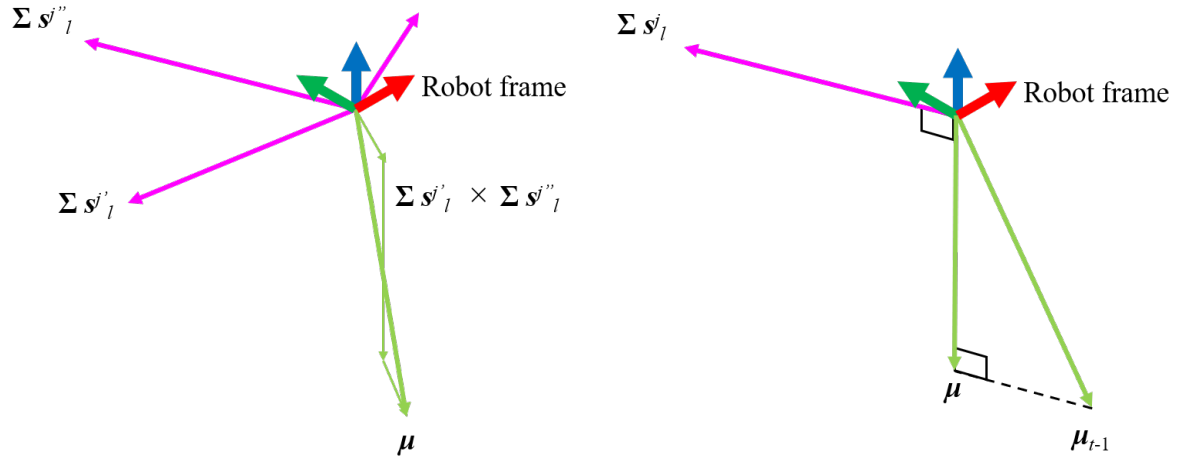
$$\boldsymbol{\mu} = \sum_{\check{j}',\check{j}''(\check{j}'\neq\check{j}'')}^{\#\check{\boldsymbol{W}}} \left(\sum_{l=0}^{\#\boldsymbol{w}_{\check{j}'}} \boldsymbol{s}_l^{\check{j}'} \times \sum_{l=0}^{\#\boldsymbol{w}_{\check{j}''}} \boldsymbol{s}_l^{\check{j}''} \right) \tag{2.24}$$

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

- $\#\check{\boldsymbol{W}} = 1$ の場合，前ステップの推定値 $\boldsymbol{\mu}_{t-1}$ を用いて重力ベクトル $\boldsymbol{\mu}$ を推定する．図 2.10b に計算例を示す．

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

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



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

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 $\boldsymbol{x}$ は，ロボットのロール ϕ とピッチ θ で構成されている．

$$\boldsymbol{x} = \begin{pmatrix} \phi & \theta \end{pmatrix}^{\mathrm{T}} \tag{2.26}$$

状態ベクトル $\boldsymbol{x}$ と共分散行列 $\boldsymbol{P}$ は，予測プロセスと更新プロセスの両方で逐次的に計算される．角速度センサの計測値は，予測プロセスで積算される（2.4.1 項）．カメラを用いた DNN の推定値と，LiDAR を用いた点群処理による推定値は，それぞれ異なる更新プロセスで観測される（2.4.2，2.4.3 項）．ここで，t は時間ステップ，S_ϕ，C_ϕ，T_ϕ 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 $\boldsymbol{x}$ と共分散行列 $\boldsymbol{P}$ は，それぞれ以下のように計算される．

$$\begin{aligned} \bar{\boldsymbol{x}}_t &= f_{(\boldsymbol{x}_{t-1},\boldsymbol{u}_{t-1})} = \boldsymbol{x}_{t-1} + \boldsymbol{Rot}^{\mathrm{rpy}}_{(\boldsymbol{x}_{t-1})}\boldsymbol{u}_{t-1} \\ \boldsymbol{u}_{t-1} &= \boldsymbol{\omega}_{t-1}\Delta t = \begin{pmatrix} \omega_{\mathrm{x}_{t-1}}\Delta t \\ \omega_{\mathrm{y}_{t-1}}\Delta t \\ \omega_{\mathrm{z}_{t-1}}\Delta t \end{pmatrix}, \quad \boldsymbol{Rot}^{\mathrm{rpy}}_{(\boldsymbol{x}_{t-1})} = \begin{pmatrix} 1 & S_{\phi_{t-1}}T_{\theta_{t-1}} & C_{\phi_{t-1}}T_{\theta_{t-1}} \\ 0 & C_{\phi_{t-1}} & -S_{\phi_{t-1}} \end{pmatrix} \end{aligned} \tag{2.27}$$

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

$$\bar{\boldsymbol{P}}_t = \boldsymbol{J}_{\boldsymbol{f}_{t-1}}\boldsymbol{P}_{t-1}\boldsymbol{J}_{\boldsymbol{f}_{t-1}}^{\mathrm{T}} + \boldsymbol{Q}_{t-1}, \quad \boldsymbol{J}_{\boldsymbol{f}_{t-1}} = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_{t-1},\boldsymbol{u}_{t-1}} \tag{2.28}$$

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

2.4.2 カメラを用いた更新プロセス

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

$$\boldsymbol{z} = \hat{\boldsymbol{\mu}}_{\mathbf{camera}} \tag{2.29}$$

ここで，$\hat{\boldsymbol{\mu}}_{\mathbf{camera}}$ 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．

$$\begin{aligned} h_{(\boldsymbol{x}_t)} &= \boldsymbol{Rot}^{\mathrm{xyz}}_{(-\boldsymbol{x}_t)}\frac{\boldsymbol{g}_{\mathbf{world}}}{|\boldsymbol{g}_{\mathbf{world}}|}, \quad \boldsymbol{g}_{\mathbf{world}} = \begin{pmatrix} 0 \\ 0 \\ g_{\mathrm{world}} \end{pmatrix} \\ \boldsymbol{Rot}^{\mathrm{xyz}}_{(\boldsymbol{x}_t)} &= \begin{pmatrix} C_{\theta_t}C_{\psi_t} & S_{\phi_t}S_{\theta_t}C_{\psi_t} - C_{\phi_t}S_{\psi_t} & C_{\phi_t}S_{\theta_t}C_{\psi_t} + S_{\phi_t}S_{\psi_t} \\ C_{\theta_t}S_{\psi_t} & S_{\phi_t}S_{\theta_t}S_{\psi_t} + C_{\phi_t}C_{\psi_t} & C_{\phi_t}S_{\theta_t}S_{\psi_t} - S_{\phi_t}C_{\psi_t} \\ -S_{\theta_t} & S_{\phi_t}C_{\theta_t} & C_{\phi_t}C_{\theta_t} \end{pmatrix} \end{aligned} \tag{2.30}$$

ここで，$\boldsymbol{g}_{\mathbf{world}}$ はワールド座標系における重力ベクトル（i.e. $g_{\mathrm{world}} \fallingdotseq 9.8\,\mathrm{m/s^2}$），$\boldsymbol{Rot}^{\mathrm{xyz}}$ 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 $\boldsymbol{R}$ は，DNN の出力 $\boldsymbol{\Sigma}$ を用いて設定される．

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

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

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

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

2.4.3 LiDAR を用いた更新プロセス

以下の観測ベクトル $\boldsymbol{z}$ を用いて，式 2.30，2.32 と同様に，状態ベクトル $\boldsymbol{x}$ と共分散行列 $\boldsymbol{P}$ をそれぞれ更新する．

$$\boldsymbol{z} = \frac{\boldsymbol{\mu}_{\mathbf{lidar}}}{|\boldsymbol{\mu}_{\mathbf{lidar}}|} \tag{2.33}$$

ここで，$\boldsymbol{\mu}_{\mathbf{lidar}}$ 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ηが推論の不確かさを表すと仮定して，閾値TH_ηによって分散の小さいサンプルを選択する．本検証では，その閾値を$\mathrm{TH}_\eta = \frac{1}{\#D}\sum_{\iota=0}^{\#D}\eta_\iota$とする．ここで，$\#D$はテストデータセットのサンプル数である．
- ‘Regression’は，FC最終層が‘MLE (ours)’とは異なるネットワークを表す．このネットワークは，共分散を含まない3次元の重力ベクトルを出力する．これは，関連研究[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$\sum_{\iota=0}^{\#D}\boldsymbol{g}_\iota$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

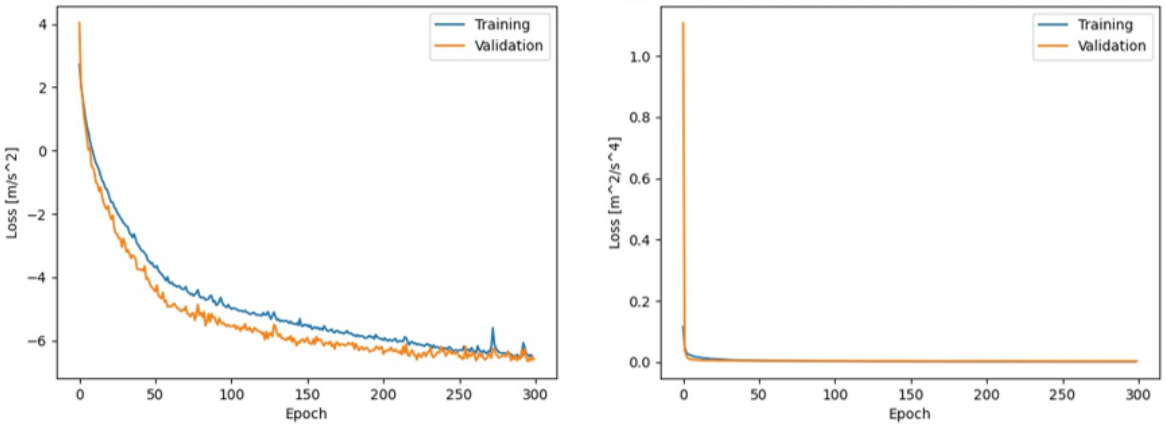
今回の検証で使用したデータセットを表3.1に示す．ネットワークは，10000個のシミュレーションデータサンプル（Dataset#1）を用いて，バッチサイズを200，エポック数を

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	10000	Training
2		‘Neighborhood’	1000	Test
3	Real	AreaI	1108	Fine-tuning
4		AreaII (daytime)	443	Test
5		AreaII (nighttime)	447	Test



(a) MLE (ours)　　(b) Regression

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.

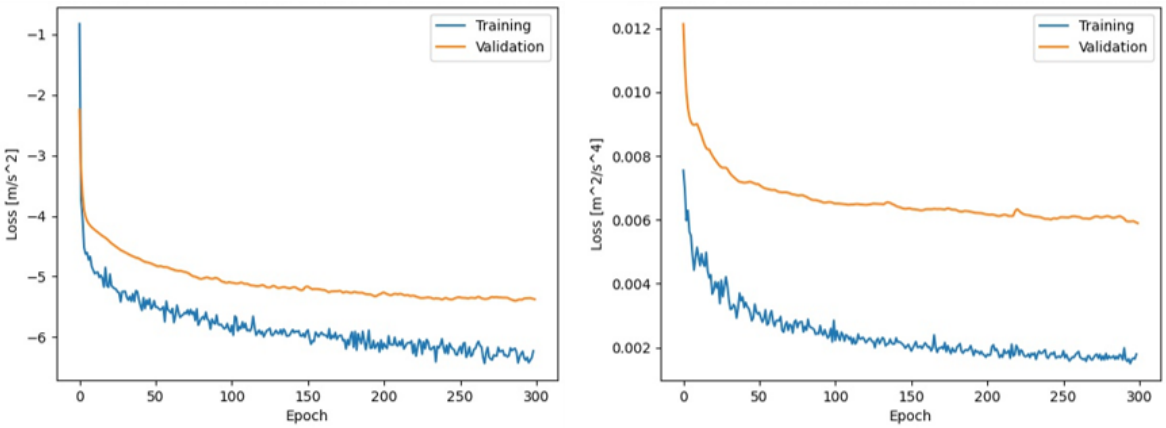
表 3.2: Loss after 300 epochs of training.

	Train (#1)	Test (#2)
MLE (ours) [m/s^2]	-6.5002	-6.5961
Regression [m^2/s^4]	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た．しかし，テストデータセットでの損失値は訓練データセットでの損失値よりも大きくなった．訓練データとテストデータでの結果の差を小さくするためには，より多様なデータを用いて学習を行う必要がある．



(a) MLE (ours)　　(b) Regression

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^2]	-6.2295	-4.4907
Regression [m^2/s^4]	0.0018	0.0105

3.1.4　静的姿勢推定

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

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

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，$\eta < \mathrm{TH}_\eta = \frac{1}{\#D}\sum_{\iota=0}^{\#D}\eta_\iota = 0.000120\ \mathrm{m}^3/\mathrm{s}^6$を満たす795個のサンプルが選択された．この閾値は他のデータセットでも使用された．この閾値を用いて選択されたサンプル数を表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，TH_ηによるフィルタリングが有効であることがわかり，提案ネットワークは共分散行列を出力することで不確かさを表現していることがわかる．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η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η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TH_η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	**1.368**
	Pitch	**0.824**	**1.121**
Regression	Roll	**1.012**	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	**1.762**	**2.265**	**2.139**
		Pitch	**4.043**	**4.590**	**4.919**
	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	**1.253**	**1.904**	**1.930**
		Pitch	**1.114**	**2.285**	**2.282**
	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

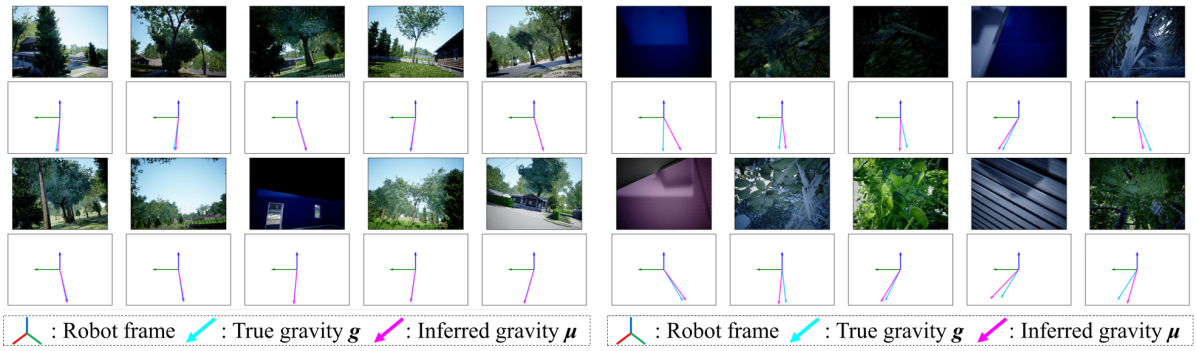
- 'Gyro+MLE' は，角速度センサで計測する角速度と，提案 DNN（図 2.7）の推論を統合する EKF ベースの推定手法を表す．推論される分散値 η が大きい場合，推論は統合されない．ハイパーパラメータは，$\mathrm{TH}_\eta = 1.2 \times 10^{-4}\ \mathrm{m}^3/\mathrm{s}^6$，$\xi = 1 \times 10^3$ と設定される．これらは，3.1 節の検証結果や経験を基に決められた．

- '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．

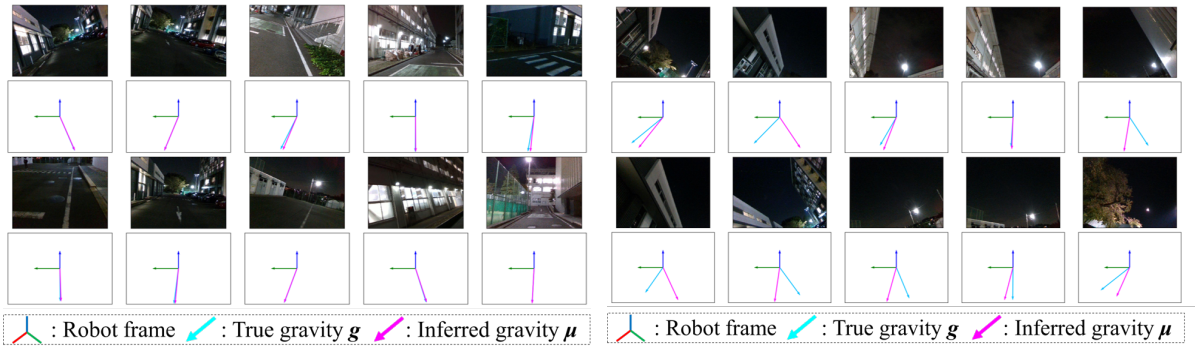
3.2.2 実験条件

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 $\mathrm{m/s^2}$，標準偏差 0.5 rad/s, 0.5 $\mathrm{m/s^2}$ 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 η (b) Top 10 largest variance η

図 3.3: Synthetic samples (Dataset#2) sorted in η values.
The DNN tends to output large error and variance when the image does not have enough landscape imformation.



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

図 3.4: Real samples (Dataset#5) sorted in η 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った．‘Gyro’ は累積誤差が大きくなった．仮想ノイズが付与されていて，他の観測がないため，これは当然のことである．‘Gyro+Acc’ は誤差を蓄積しなかった．しかし，センサの加速度値にはロボット自身の加速度やノイズが含まれているため，常に誤差が発生した．一方，提案手法では，それらを含まない重力ベクトルを観測することができる．‘Gyro+NDT’ は LiDAR を用いることで，‘Gyro’ よりゆっくり誤差を蓄積したが，累積誤差を取り除くことはできなかった．

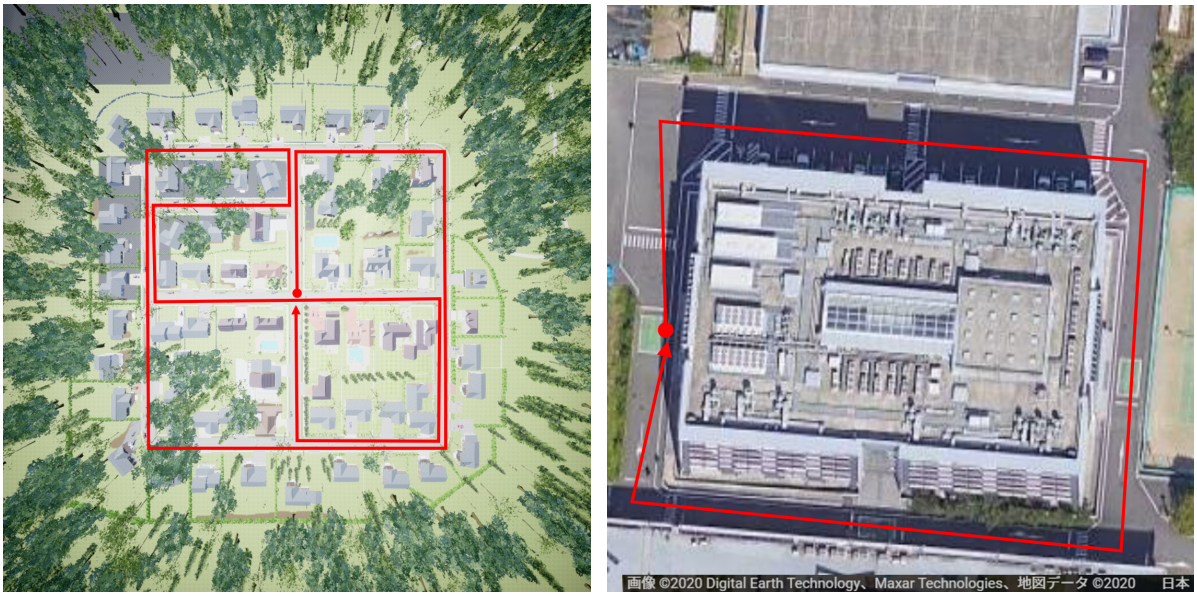
表 3.7: Hyperparameters.

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

'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，η が大きい DNN の出力を棄却することが有効であることがわかった．'Gyro+MLE' では，飛行中に，閾値 TH_η によって約 13 %の推論が棄却された．これにより，不確実性の高い出力の観測を回避することができた．特に，'MLE' では，姿勢角が訓練範囲 [-30 deg, 30 deg] を超えると，分散 η が大きくなる傾向があった．一方で，推論を棄却することで，誤差を修正する機会は減る．この機会の減少を補うために，提案手法は，カメラを用いた重力方向推定と，LiDAR を用いた重力方向推定の両方を統合している．飛行中，予測プロセス（2.4.1 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)	**2.772**	**2.141**



(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．

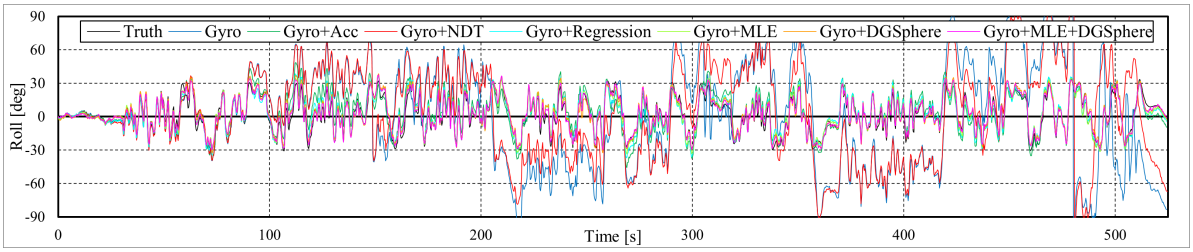
3.3.1 モーションキャプチャを用いた屋内実験

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．

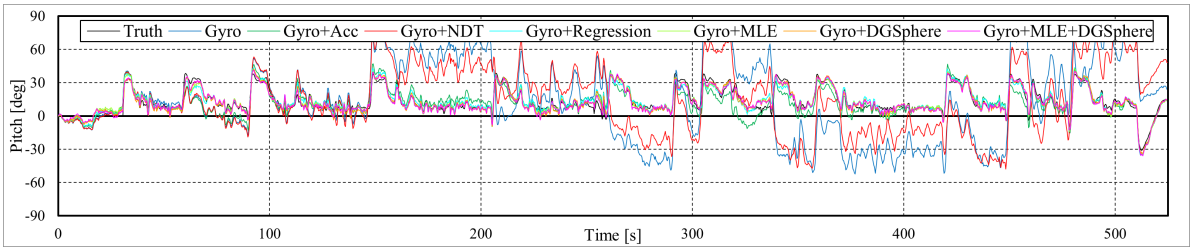
表 3.9 に推定姿勢の MAE を示す．提案手法は，実世界においても誤差の蓄積を抑制することができた．平坦な室内環境では，提案手法の MAE は，‘Gyro+Acc’ の MAE とほぼ同じであった．振動などがより発生する環境では，‘Gyro+Acc’ の精度はより低くなると予想でき [30]，それは前節のシミュレーションで再現された．

3.3.2 屋外実験

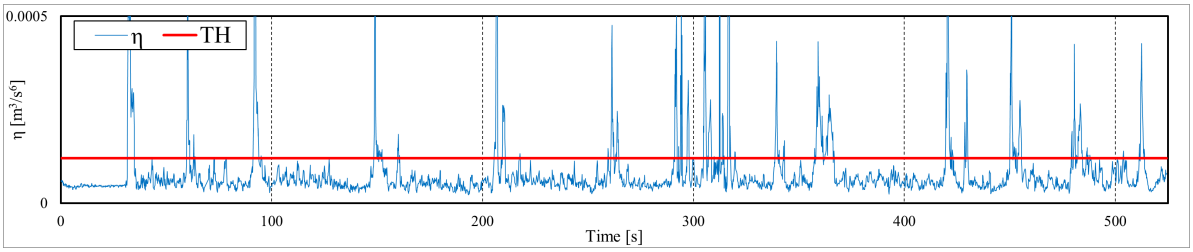
モーションキャプチャカメラは姿勢を正確に測定できるが，キャプチャできる範囲が限られている．それを補うために，長距離移動実験も行った．詳細な定量評価は前項で行ったので，本項は，提案手法が実世界でも動作することを確認するためのものである．



(a) Roll



(b) Pitch



(c) η

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 η 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 $\phi_{\mathrm{gt}} = 0$ deg, $\theta_{\mathrm{gt}} = 0$ deg と仮定した．この評価方法は，関連研究 [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	**1.648**
Gyro+Regression	2.843	2.730
Gyro+MLE	2.367	2.003
Gyro+DGSphere	2.365	1.878
Gyro+MLE+DGSphere (ours)	**2.242**	1.839

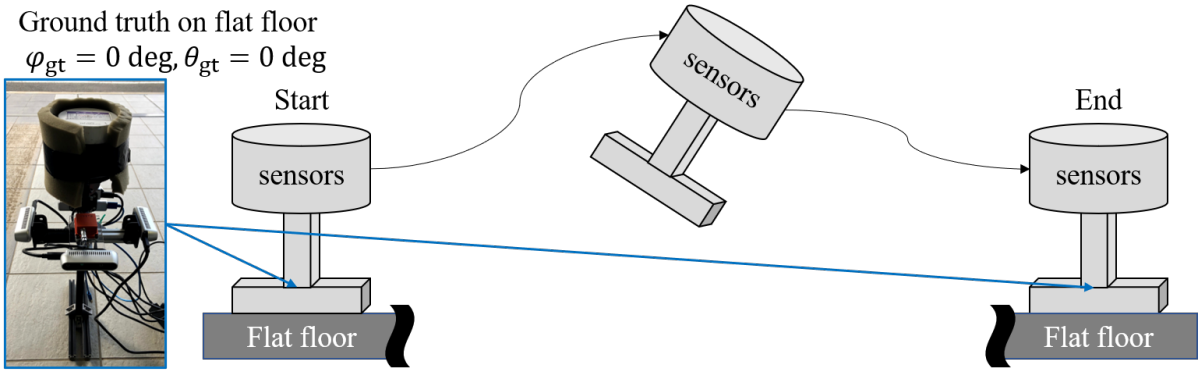


図 3.7: Dynamic experiment.
The ground truth on the flat floor is assumed to be $\phi_{gt} = 0$ deg, $\theta_{gt} = 0$ deg.

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

	Roll [deg]	Pitch [deg]
Gyro+MLE+DGSphere (ours)	**+0.515**	**+2.110**
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せることが示された．

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 fi ltering 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. *Procceding 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 録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

- 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

- 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

- 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

付 録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.