LiDAR-Inertial SLAM 入門

赤井 直紀[*1]

2025 年 9 月 29 日

[*1] 名古屋大学，株式会社 LOCT

まえがき

私は 2011 年から研究室に所属し，そこから移動ロボットの自動走行に関する研究に携わってきました．当時は 3D LiDAR がメジャーではなく，LiDAR といえば 2D LiDAR のことを指していました．そして 2D LiDAR を用いた自己位置推定方法としては，Monte Carlo Localization（MCL）[**?**] が主流でした．MCL はパーティクルフィルタベースの手法であり，枠組みとしては確率論的アプローチの手法になります．この手法は実装が容易，かつ拡張性も十分にあるため，私はこれまで，基本的に MCL を用いて確率的な自己位置推定に関する研究を主に行ってきました．特に 2D LiDAR を軸として，新しい枠組みのようなものを提案しようと取り組んできました．手前味噌で恐縮ですが，一応私の研究の代表作としては以下の論文があります [**?**].

しかし世の中的には 3D LiDAR が一般的になっていました．そんなある日の 2023 年頃，とある会社の方からご相談を受け，「3D LiDAR を使った LIO が作れないか」と尋ねられました．LIO の名前はもちろん知っていましたが，あまり LIO に関する情報を追いかけていなかったので，ひとまず有名な FAST LIO [**?**] の論文を読んでみました．そして論文を読んでみた最初の感想が「わからん」でした．リー群やリー代数の知識が当たり前のように使われていて，それらがわからない人からすると全く何を言っているかわからないという感じでした．しかし FAST LIO を用いてできていることはものすごいことだなと実感したので，頑張って読み解いてみようと思い，約半年ぐらいかけて最低限の内容は理解し，LIO の実装などができるようになってきました．

しかし LIO が実装できただけではきれいな地図は作れず，そこからリー群を用いたグラフベース SLAM などの実装なども行いました．結果として，1 年以上かけてリー群を用いた LIO や SLAM の基礎は最低限得られたかなと感じました．そしてこれらを通して得られた経験を基に，一度 LIO や SLAM の機能を整理して実装しなおそうとしたのが *plain_slam_ros2* になります[*1]．LIO，特に SLAM のソフトウェアは，基本的にシステムが膨大になってしまい，なかなかソフトの中身を見てみようと思えないかと思いますが，*plain_slam_ros2* は比較的にコンパクトに構成をまとめることができたと思っています．そして *plain_slam_ros2* を作成した後に，「これを作成するために使用した知見をすべてまとめ，今後 LiDAR SLAM の勉強を行う方の参考にして貰いたい」と思い，本書[*2]を作成することを決意しました．

本書を理解するためには，以下の数学的知識が前提となります．

- 線形代数（行列演算，回転行列，剛体変換）
- 微分積分（多変数関数の偏微分，ヤコビアン）

また，以下の知識があると理解がより深まります．

- 確率・統計（ガウス分布，最尤推定）
- 数値最適化（非線形最小二乗法）
- リー群・リー代数（SO(3)，SE(3) とその指数写像・対数写像）

ただし本書を理解する上で最低限必要な数学的な知識は 2 章で述べています．

[*1] `https://github.com/NaokiAkai/plain_slam_ros2`

[*2] 「本書」というように，一応本という体で述べていますが，第 3 者からのチェック等は受けていないため，間違っている部分もあるかもしれないことを前提に読み進めてください．

第 1 章

はじめに

1.1 背景知識

ロボットの自律移動や自動車の自動運転を行うにあたっては，地図を構築し，その地図上で走行している位置を認識する技術，いわゆる**自己位置推定**（Localization）や **Simultaneous Localization and Mapping**（SLAM）とよばれる技術が重要であるとされています [**?**]．従来，これらの技術を用いる場合は，**オドメトリ**（Odometry）と呼ばれる移動量推定を行う枠組みが用いられていました．自己位置推定や SLAM で用いられている技術を端的に述べると，構築された地図（SLAM の場合はオンラインで構築している地図）と，センサの観測値を照合することで，地図上のどの位置に自分が存在しているかを認識する技術になります．この「センサの観測値と地図の照合」を行うにあたり，オドメトリから予測された移動量を用いることで，どの程度移動したかを予測することが可能となり，照合を行う際の探索範囲を限定することができるようになります．そのためオドメトリを用いると，自己位置推定および SLAM の精度や頑健性を向上させることができます．

しかしオドメトリを用いるとなると，自己位置推定や SLAM で用いられる外界センサ（LiDAR やカメラ）以外のセンサが必要になります [**?**]．オドメトリシステムを構築する上で最も簡単な方法（システムを構築する手間が最もかからないという意味で）は，**Inertial Measurement Unit**（IMU）を用いることだといえます．IMU は，センサを原点とした加速度と角速度を計測できるセンサであり，これらの値を積分していくだけで移動量を計算することができます．しかし IMU の計測値が含む誤差は大きく，単に積分して得られた位置や角度の精度は極めて低く，自己位置推定や SLAM には利用できないことがほとんどです．そのため，IMU だけを用いてオドメトリシステムを構築することは不可能に近いといえます．

移動ロボットや自動運転の分野で最も広く使われているオドメトリシステムは，エンコーダ等を用いて車輪の回転量を計測し，その結果を積分することで移動量を計算する方法です．これは**ホイールオドメトリ**（Wheel Odometry）と呼ばれます．ホイールオドメトリを用いれば，タイヤの空転等が発生しない限り，短距離であれば移動量を正確に計測することができます．ただしホイールオドメトリを用るには，外界センサだけでなく，移動体のハードウェアにも大きな変更を加える必要がります．そのためホイールオドメトリは，安易に利用できるシステムとは言い難いです．またホイールオドメトリは，車輪の回転量を計測することが前提のため，基本的に車輪型の移動体にしか適用することができません．

ホイールオドメトリに頼らない移動量の推定方法として，**ビジュアルオドメトリ**（Visual Odometry）が提案されました [**?**]．ビジュアルオドメトリとは，画像から得られる特徴を追跡することで移動量を推定する方法です．そのためビジュアルオドメトリは，車輪型以外の移動体にも適用することが可能で

す．しかし一般に，ビジュアルオドメトリの精度はホイールオドメトリ程高くはないことが知られています．

同様に LiDAR を用いて移動量推定を行う方法も様々研究されていましたが，本格的に **LiDAR Odometry**（LO）という言葉が使われた始めた代表的な手法として LiDAR Odometry and Mapping（LOAM）があります [**?**]．LO では，LiDAR が計測する点群を逐次的に照合していくことで移動量の推定を行います．LiDAR の距離計測の精度は高いため，LO による移動量推定の精度は高くなります．そのため LO は，様々な用途で使われるようになりました．

しかし LO にも弱点がありました．LiDAR，特に 3D LiDAR の計測周期は遅く（一般に 10 ∼ 20 Hz 程度），高速な動き，特に回転を含む移動量を推定することは困難でした．この問題を解決する方法として提案されたのが，LiDAR と IMU を融合して移動量推定を行う **LiDAR–Inertial Odometry**（LIO）です．IMU は高周期（一般に 100 Hz 以上）で加速度と角速度を計測することができるため，LiDAR の計測周期の移動量を補間することができます．この移動の補間を用いると，高速に移動する LiDAR によって歪んでしまった LiDAR の計測点群を補正することができるようになります．また LIO では，LO では推定されていなかった状態量（速度や IMU の計測値のバイアス）の推定も行います．そのため，IMU の計測値の積分も正確に行えるようになるため，移動量の推定をより高精度に行うことが可能になりました．なお Vision と IMU を融合した **Visual–Inertial Odometry**（VIO）は，LIO よりも少し早く提案されていました（例えば [**?**]）．

1.2　LIO の性能と限界

LIO のアルゴリズムの進化は目覚ましいものがありましたが，LiDAR の性能自体もここ数年で大きく進化しました．一昔前（著者が研究を始めたのが 2011 年）では，「LiDAR は価格コストが高いため，カメラを用いた手法を提案する」というのが論文等では常套文句でした．しかし今では，日本円で 10 万円程度で購入可能な 3D LiDAR も発売されています．そして驚くことに，このような価格の 3D LiDAR でも，360 度 100 m に近いレンジを計測できるようになっており，LIO を用いて高精度な移動量推定を行うということはかなり一般的になってきました．そして LIO を用いるだけでも，小規模な環境であれば十分な精度の点群地図を構築することができるようになりました．そのため，ドローンのような飛翔体にこのような小型の LiDAR を搭載し，高精度な点群地図を生成することも容易に行われるようになってきました．

ただし LIO はあくまでオドメトリシステムであるため，移動量推定しか行いません．そのため，どれだけ高精度に移動量が推定できたとしても，推定量に誤差（ドリフト）が含まれてしまうため，LIO だけを用いて大規模な環境の地図構築を行うことは依然として難しいです．特に大規模でループ（一度通過した地点を再度通過すること）が含まれる環境や条件ですと，整合性の取れた地図が構築できなくなってしまいます（同じものが同じ地点に正しくマッピングされなくなります）．前述した SLAM では，このようなループが含まれる場合であっても，整合性が取れた地図構築を行うことを目的としています．すなわち，精度の高い地図を構築したい場合には，SLAM の利用は避けられません．

また LIO はあくまで移動量推定のシステムです．応用上においては，移動量がわかるだけでは嬉しさがあることは少ないといえ，SLAM で構築した地図上で，どの位置にいるかを知れることの方が恩恵が多いといえます．例えば工場等で AGV やフォークリフトの位置を管理したい場合などには，LIO の利用だけでは不十分であり，自己位置推定の利用が求められます．そのため，単に高精度の LIO が利用可能になったというだけでは新たな応用システムを提案することは難しく，LIO に含まれるアルゴリ

ズムを正しく理解し，それを SLAM や自己位置推定にも応用していくことが重要になります．

1.3 既存手法と本書の立ち位置

LIO や SLAM を行うオープンソースはすでに多数存在しています．例えば LIO の有名なオープンソースとしては LIO-SAM [**?**] や FAST-LIO [**?**] が挙げられます．これらの手法の性能は極めて高く，これらをダウンロードして使用するだけでも，十分な移動量推定を行うことができるといえます．また LIO-SAM には SLAM の機能も含まれているため，地図構築を行うこともできます．また LiDAR SLAM の有名なオープンソースとしては，GLIM [**?**] が挙げられます．これらの性能も極めて高く，様々な環境で極めて精度の高い点群地図を構築することができます．

しかし多くのソースコードは，機能を多く含むため，どうしても規模が大きくなってしまいます．そのため，初めて SLAM を学ぼうとする人がこれらを見ても，どこから何を追えば良いかの判断が難しく，結局ダウンロードして使うだけになってしまうことが多いと思います．本書，および対応するソースコードは，ソフトウェアの構成をとにかくシンプルに実装することに重きをおいています．開発したソースコードは，LIO，SLAM，自己位置推定の機能を有しており，その主な処理は空行を除いて 2000 行未満の C++ で完結しています．この C++ の中に，スキャンマッチング，LiDAR と IMU の融合（ルーズカップリングとタイトカップリング），ループ検知，ポーズグラフの最適化といった必要な処理をすべて実装しています（用語の詳細は後ほど解説します）．また依存ライブラリも極力少なくし，ほぼフルスクラッチで LIO や SLAM を実装できるようになるようにしています．なお，主な依存ライブラリは *Sophus*（*Eigen* ベース）と *nanoflann* のみになります（他はパラメーター設定のために *YAML* を用いています）．これらはそれぞれ線形・リー代数を扱うライブラリと，最近某探索を行うライブラリとなっており，LIO や SLAM の根幹となる部分はすべてフルスクラッチで実装されているため，初学者でもどのように最適化などが実装されているか理解しやすい構成になっていると思います．

第 2 章

数学的知識

2.1 表記

本書では基本的に実数しか扱いません．そのため断りのない限り，実数が使われていることを前提とします．表記としては，スカラーを $a \in \mathbb{R}$，N 次元のベクトルを $\mathbf{a} \in \mathbb{R}^N$，$N \times M$ の行列を $A \in \mathbb{R}^{N \times M}$ と表記します．また単位行列をよく用いますが，次元をわかりやすくするために添え字を付けます．例えば N 次元の単位行列は I_N と示します．

2.2 ヤコビアン

本書では主に**最適化**（Optimization）を利用していきます．最適化とは，ある変数 $\mathbf{x} = (x_1 \ \cdots \ x_N)^\top \in \mathbb{R}^N$ に従う関数 $f(\mathbf{x})$（最適化に使われる関数はよく**コスト関数**（Cost Function）と呼ばれます）が定義されたときに，f の値を最小化（もしくは最大化）させること，またそれに対応する変数 $\mathbf{x}$ を求めることになります．最適化を行う方法には様々な方法がありますが，基本となることは，関数の勾配を求めることになり，これは式 (2.1) で定義されます．

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f(\mathbf{x})}{\partial x_N} \end{pmatrix} \tag{2.1}$$

なお，関数を引数であるベクトルで偏微分した結果は，**ヤコビアン**（Jacobian）と呼ばれます．ベクトル関数 $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}) \ \cdots \ f_M(\mathbf{x}))^\top \in \mathbb{R}^M$ に関してもヤコビアンを定めることができ，これは式 (2.2) で表すことができます．

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_M(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_M(\mathbf{x})}{\partial x_N} \end{pmatrix} \tag{2.2}$$

なお本書では，基本的にヤコビアンを J と表記します．

ヤコビアンの計算は基本的に式 (2.1)，(2.2) に示す定義に従って行いますが，少し異なった方法でも導出することが可能であり，本書では基本的にこの方法を用いてヤコビアンを求めていきます．まず，関数 $f(\mathbf{x})$ を $\delta\mathbf{x}$ だけ変化させた結果をテイラー展開を用いて近似します．

$$f(\mathbf{x} + \delta\mathbf{x}) \simeq f(\mathbf{x}) + J\delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^\top H \delta\mathbf{x} \tag{2.3}$$

なお $H = \partial^2 f(\mathbf{x}) / \partial \mathbf{x}^2$ であり，これは**ヘッセ行列**（Hessian）と呼ばれます．ここで，2 次の微小量を

無視し，式 (2.3) の両辺が等しいと仮定すると，次式が得られます．

$$f(\mathbf{x}+\delta\mathbf{x}) - f(\mathbf{x}) = J\delta\mathbf{x} \tag{2.4}$$

すなわち，$f(\mathbf{x}+\delta\mathbf{x})$ と $f(\mathbf{x})$ の差分を $J\delta\mathbf{x}$ という形で記述できるとヤコビアンを求めることができます．

例えば，$f(x) = x^2$ という簡単な例を考えてみます．この関数のヤコビアン[*1]は $\partial x^2/\partial x = 2x$ となりますが，式 (2.4) に示す方法でヤコビアンを求めてみます．

$$\begin{aligned}(x+\delta x)^2 - x^2 =& x^2 + 2x\delta x + \delta x^2 - x^2 \\ =& 2x\delta x\end{aligned} \tag{2.5}$$

ただし，$\delta x^2 \simeq 0$ として 2 次の微小量を無視しました．式 (2.5) に示すように，$f(x) = x^2$ のヤコビアンを正しく求めることができました．この方法を用いると，関数を直接微分しなくても，ヤコビアンを求めることができます[*2]．

2.3　ガウス・ニュートン法

本書で扱う問題は，度々最適化問題に帰着されます．最適化問題を解く際に用いられる方法は様々ありますが，本書では**ガウス・ニュートン法**（Gauss-Newton Method）を主に用います．

ガウス・ニュートン法を考えるにあたり，まず状態変数 $\mathbf{x} \in \mathbb{R}^N$，およびこの状態に依存する**残差**（Residual）（もしくは残差ベクトル）$\mathbf{r}(\mathbf{x}) \in \mathbb{R}^M$ を導入します．今，複数の残差ベクトルを用いて，以下のコスト関数を定義します．

$$E(\mathbf{x}) = \sum_i \|\mathbf{r}_i(\mathbf{x})\|_2^2 \in \mathbb{R} \tag{2.6}$$

ここで $\|\cdot\|_2^2$ は，ベクトルのユーグリッドノルムの 2 乗を計算する操作です．そして，コスト関数を最小化する状態を以下のように定義します．

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} E(\mathbf{x}) \tag{2.7}$$

これは，$\mathbf{x}$ の定義域においてコスト関数 $E(\mathbf{x})$ を最小にするような状態 $\mathbf{x}^*$ を求めることを意味します．このような解は，**最適解**（Optimal Solution）と呼ばれます．

ガウス・ニュートン法による最適化を考えるにあたり，まず残差ベクトルを 1 次のテイラー展開で近似することを考えます．

$$\mathbf{r}(\mathbf{x}+\delta\mathbf{x}) \simeq \mathbf{r}(\mathbf{x}) + J\delta\mathbf{x} \tag{2.8}$$

ここで J は残差ベクトル $\mathbf{r}$ の状態ベクトル $\mathbf{x}$ に関するヤコビアン $\partial\mathbf{r}/\partial\mathbf{x} \in \mathbb{R}^{M\times N}$ になります．次に，式 (2.8) の近似された残差ベクトルの 2 乗ノルムを考えます．

$$\begin{aligned}(\mathbf{r}+J\delta\mathbf{x})^\top(\mathbf{r}+J\delta\mathbf{x}) =& \left(\mathbf{r}^\top + \delta\mathbf{x}^\top J^\top\right)(\mathbf{r}+J\delta\mathbf{x}) \\ =& \mathbf{r}^\top\mathbf{r} + \mathbf{r}^\top J\delta\mathbf{x} + \delta\mathbf{x}^\top J^\top\mathbf{r} + \delta\mathbf{x}^\top J^\top J\delta\mathbf{x} \\ =& \mathbf{r}^\top\mathbf{r} + 2\delta\mathbf{x}^\top J^\top\mathbf{r} + \delta\mathbf{x}^\top J^\top J\delta\mathbf{x}\end{aligned} \tag{2.9}$$

[*1] 正確には引数がスカラーなので導関数と呼ぶのが一般的ですが，1 次元の場合のヤコビアンと形式的に見なすことができます．

[*2] ここで得られるヤコビアンは，関数をテイラー展開し，2 次以上の微小項を無視することで得られる 1 次近似により得られたものです．$\delta x \to 0$ の極限では，理論的な微分（導関数）と一致しますが，有限の差分を用いる場合は数値的な近似値となります．

なお，$\mathbf{r}^\top J\delta\mathbf{x} = \delta\mathbf{x}^\top J^\top \mathbf{r}$ を用いています．

続いて，式 (2.9) を $\delta\mathbf{x}$ の関数 $f(\delta\mathbf{x})$ とみなして，$\delta\mathbf{x}$ で偏微分します．

$$\frac{\partial f(\delta\mathbf{x})}{\partial \delta\mathbf{x}} = 2J^\top \mathbf{r} + 2J^\top J\delta\mathbf{x} \tag{2.10}$$

そして，式 (2.10) が $\mathbf{0}$ になると仮定すると，次式が成り立ちます．

$$J^\top J\delta\mathbf{x} = -J^\top \mathbf{r} \tag{2.11}$$

ここで，式 (2.11) を満たす $\delta\mathbf{x}$ について考えます．$f(\delta\mathbf{x})$ は，$\delta\mathbf{x}$ だけ状態を変化させたときの残差ベクトルを近似し，その 2 乗ノルムを計算したものになっています．これを $\delta\mathbf{x}$ に関して偏微分し，その結果を $\mathbf{0}$ とすると，近似した残差ベクトルの 2 乗ノルムを最小にする $\delta\mathbf{x}$ を求めることが可能になります．すなわち，この操作で求められた $\delta\mathbf{x}$ 分だけ $\mathbf{x}$ を更新すると，コスト関数を減少させることができるようになります．

式 (2.9) を導出するにあっては，1 つの残差ベクトルの近似を考えましたが，実際のコスト関数では複数の残差ベクトルの 2 乗ノルムの和を計算しています．そのため，状態 $\mathbf{x}$ を $\delta\mathbf{x}$ だけ変化させたコスト関数を近似する必要がありますが，これは式 (2.12) のようになります．

$$E(\mathbf{x}+\delta\mathbf{x}) \simeq \sum_i \left(\mathbf{r}_i^\top \mathbf{r}_i + 2\delta\mathbf{x}^\top J_i^\top \mathbf{r}_i + \delta\mathbf{x}^\top J_i^\top J_i \delta\mathbf{x}\right) \tag{2.12}$$

そして同様に，式 (2.12) を $\delta\mathbf{x}$ で偏微分した結果を $\mathbf{0}$ にすると，次式が得られます．

$$\sum_i J_i^\top J_i \delta\mathbf{x} = -\sum_i J_i^\top \mathbf{r}_i \tag{2.13}$$

ここで簡略化のため，以下のように変数を導入します．

$$\begin{aligned} H &= \sum_i J_i^\top J_i \\ \mathbf{b} &= \sum_i J_i^\top \mathbf{r}_i \end{aligned} \tag{2.14}$$

これにより式 (2.13) は $H\delta\mathbf{x} = -\mathbf{b}$ と書けます．なおこの H と $\mathbf{b}$ はそれぞれヘッセ行列と勾配とも呼ばれます．ガウス・ニュートン法を用いた最適化では，$H\delta\mathbf{x} = -\mathbf{b}$ を満たす $\delta\mathbf{x}$ を得た後に，以下のように状態を更新します．

$$\mathbf{x} \leftarrow \mathbf{x} + \delta\mathbf{x} \tag{2.15}$$

なお，$H\delta\mathbf{x} = -\mathbf{b}$ からは，当然 $\delta\mathbf{x} = -H^{-1}\mathbf{b}$ が導けますが，$H \in \mathbb{R}^{N\times N}$ になるため，N が大きい場合には逆行列の直接的な計算が困難になります．そのため，直接逆行列を計算することが少ないため，「$H\delta\mathbf{x} = -\mathbf{b}$ を満たす $\delta\mathbf{x}$」という表現を用いています．N が大きくない場合は，直接 H^{-1} を計算しても実用上は問題になりません．

2.4 ロバストカーネル

最適化を実行するにあたり，残差ベクトルを複数定義しますが，誤った対応，すなわち誤対応を用いて残差ベクトルを定義してしまうと，最適化に悪影響を及ぼすことになります．通常，誤対応によって生じる残差ベクトルのノルムは，正しい対応に比べて大きくなる傾向があります．そのため，残差ベク

トルのノルムの大きさに応じて最適化への影響を抑制することが有効です．このような効果を実現するために，**ロバストカーネル**（Robust Kernel）を導入することができます．

一般にロバストカーネルを導入したコスト関数は以下のように記述されます．

$$E\left(\mathbf{x}\right)=\sum_{i}\rho\left(\left\|\mathbf{r}_{i}\left(\mathbf{x}\right)\right\|_{2}^{2}\right) \tag{2.16}$$

式 (2.16) に示す $\rho\left(\cdot\right)$ がロバストカーネルになります．ロバストカーネルにも様々なものがありますが，本書では**フーバー損失**（Huber Loss）を用います．フーバー損失は以下のように定義されます．

$$\rho\left(s\right)=\begin{cases}s & \text{if } s\leq\delta^{2}\\ 2\delta\sqrt{s}-\delta^{2} & \text{otherwise}\end{cases} \tag{2.17}$$

ここで δ は任意の正の実数です．

フーバー損失を用いたガウス・ニュートン法を考えるにあたり，式 (2.9) に示したように，残差ベクトルの微小変化 $\mathbf{r}\left(\mathbf{x}+\delta\mathbf{x}\right)$ を線形近似した結果，すなわち $\mathbf{r}\left(\mathbf{x}\right)+J\delta\mathbf{x}$ に対するフーバー損失を考えます．

$$\rho\left(\left(\mathbf{r}+J\delta\mathbf{x}\right)^{\top}\left(\mathbf{r}+J\delta\mathbf{x}\right)\right)=\rho\left(\mathbf{r}^{\top}\mathbf{r}+2\delta\mathbf{x}^{\top}J^{\top}\mathbf{r}+\delta\mathbf{x}^{\top}J^{\top}J\delta\mathbf{x}\right) \tag{2.18}$$

ここで $s=\mathbf{r}^{\top}\mathbf{r}+2\delta\mathbf{x}^{\top}J^{\top}\mathbf{r}+\delta\mathbf{x}^{\top}J^{\top}J\delta\mathbf{x}$ とします．式 (2.17) より，$s\leq\delta^{2}$ の場合は s がそのまま用いられるため，s が δ^{2} より大きくなる例を考えることとし，与えられた s をフーバー損失に代入した結果を $\delta\mathbf{x}$ で微分します．

$$\begin{aligned}\frac{\partial\rho\left(s\right)}{\partial\delta\mathbf{x}}=&\frac{\partial\rho\left(s\right)}{\partial s}\frac{\partial s}{\partial\delta\mathbf{x}}\\ =&\frac{\delta}{\sqrt{s}}\left(2J^{T}\mathbf{r}+2J^{\top}J\delta\mathbf{x}\right)\end{aligned} \tag{2.19}$$

そして式 (2.19) が $\mathbf{0}$ になるとすると，以下が得られます．

$$\frac{\delta}{\sqrt{s}}J^{\top}J\delta\mathbf{x}=-\frac{\delta}{\sqrt{s}}J^{T}\mathbf{r} \tag{2.20}$$

式 (2.11) と比較すると，式 (2.20) では両辺に $\delta/\sqrt{s}$ が表れます．なお式 (2.20) における $\delta/\sqrt{s}$ は削除することができますが，実際には複数の残差ベクトルの和を考えることとなり，この値は考える残差ベクトル毎に異なる値となるため，削除せずに記述しています．

そして，式 (2.14) に示したように，すべての残差ベクトルを考慮してヘッセ行列と勾配を定めると以下のようになります．

$$\begin{aligned}H=&\sum_{i}\rho'\left(s_{i}\right)J_{i}^{\top}J_{i}\\ \mathbf{b}=&\sum_{i}\rho'\left(s_{i}\right)J_{i}^{\top}\mathbf{r}_{i}\end{aligned} \tag{2.21}$$

なお $\rho'\left(s\right)=\partial\rho\left(s\right)/\partial s$ であり，これは式 (2.17) より以下となります．

$$\rho'\left(s\right)=\begin{cases}1 & \text{if } s\leq\delta^{2}\\ \frac{\delta}{\sqrt{s}} & \text{otherwise}\end{cases} \tag{2.22}$$

ただし $s=\mathbf{r}^{\top}\mathbf{r}+2\delta\mathbf{x}^{\top}J^{\top}\mathbf{r}+\delta\mathbf{x}^{\top}J^{\top}J\delta\mathbf{x}$ としたため，このままで $\rho'\left(\cdot\right)$ が $\delta\mathbf{x}$ に依存してしまい，ガウス・ニュートン法の線形構造が崩れて式 (2.13) に示すような形で状態の更新量を求められなくなってしまいます．そこで $s_{0}=\mathbf{r}^{\top}\mathbf{r}$，$\delta s=2\delta\mathbf{x}^{\top}J^{\top}\mathbf{r}+\delta\mathbf{x}^{\top}J^{\top}J\delta\mathbf{x}$ とし，$\rho'\left(s\right)\simeq\rho'\left(s_{0}\right)+\rho''\left(s_{0}\right)\delta s$ と

して近似します．そして，$\rho''(s_0)\,\delta s$ が高次の微小量であるとみなして無視してしまい $\rho'(s) \simeq \rho'(s_0)$ と近似してしまいます．これにより，$\rho'(s) \simeq \rho'(s_0)$ とすることができ，$\delta\mathbf{x}$ に依存しない形となるため，ガウス・ニュートン法における線形構造を維持することができます．

最終的に，$w = \rho'\left(\mathbf{r}^\top\mathbf{r}\right)$ とする重みを定義し，ヘッセ行列と勾配を以下のように定めます．

$$\begin{aligned} H &= \sum_i w_i J_i^\top J_i \\ \mathbf{b} &= \sum_i w_i J_i^\top \mathbf{r}_i \end{aligned} \tag{2.23}$$

これらの H と $\mathbf{b}$ を用いて $H\delta\mathbf{x} = -\mathbf{b}$ を満たす $\delta\mathbf{x}$ を求め，式 (2.15) に基づき状態更新を行うことで，フーバー損失を用いたガウス・ニュートン法による最適化を実行することができます．フーバー損失を用いた場合のガウス・ニュートン法による状態更新の導出は少し複雑にはなりますが，実装にあたっては，単に式 (2.22) に示す関数を重みとして計算してそれぞれのヘッセ行列と勾配に掛けるだけなので，非常に容易に実装することができます．また多くの場合，フーバー損失を用いることで最適化がよりロバストになります．

2.5　リー群とリー代数

本書では頻繁に**リー群**（Lie Group）と**リー代数**（Lie Algebra）を用います．リー群やリー代数の厳密な説明は行いませんが，本書でリー群と呼んだ場合は**回転行列**（Rotation Matrix）と**剛体変換行列**（Rigid Transformation Matrix）を示すこととします．

回転行列は以下のように定義されます．

$$\{R \in \mathbb{R}^{3\times3} | R^\top R = I, \det(R) = 1\} \tag{2.24}$$

回転行列は Special Orthogonal Group in 3 Dimensions（SO(3)）とも呼ばれ，3 次元空間の回転を表現することができます．剛体変換行列は以下のように定義されます．

$$\left\{ \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \in \mathbb{R}^{4\times4} | R \in \mathrm{SO}(3), \mathbf{t} \in \mathbb{R}^3 \right\} \tag{2.25}$$

剛体変換行列は Special Euclidean Group in 3 Dimensions（SE(3)）とも呼ばれ，3 次元空間での回転を含む位置（姿勢）を表現することができます．LIO や SLAM では基本的に，SO(3) や SE(3) を用いて状態を表現します．なお式 (2.25) から明らかなように，T を構成する変数は $\mathbf{t}$ と R になります．そのため，本書ではしばしば $T = (R \mid \mathbf{t})$ という簡略表記を用います．

リー群を用いる利点は，回転の状態を，途中で急に値が飛んだり切り替わったりすることなく，滑らかにかつ数学的に自然な形で扱えるということです．詳細は省きますが，このような空間を**多様体**（Manifold）と呼びます．直感的には少し難しく感じるかもしれませんが，まずは平面上の回転，つまり xy 平面での角度 θ を例に考えてみます．角度 θ は通常，$0 \leq \theta < 2\pi$（あるいは $-\pi \leq \theta < \pi$）の範囲で定義されますが，$\theta = 0$ と $\theta = 2\pi$ は，数値としては異なるものの，回転としては同じ状態を表しています．このような性質から，角度 θ による表現では，状態が不連続に見えることがあります．しかしリー群を使えば，回転の変化を「切れ目のない空間」で表現でき，最初から最後まで滑らかに（連続的に）扱えるようになります．また，加減算や微分といった操作も数学的に統一されたルールで行えるので，処理が一貫してスムーズになります．しかしリー群を用いて状態を表すと，処理が少し直感的なものではなくなってしまいます．

例えばロボットの状態をあるベクトル空間で $\mathbf{x}$ と表すとします．もしロボットの状態が $\Delta\mathbf{x}$ だけ変化したとすると，直感的にはロボットの状態は $\mathbf{x}+\Delta\mathbf{x}$ という，シンプルな加算により求めることができます（ただし前述のような角度 θ が含まれる場合は値の範囲に中止いなければなりません）．しかし式 (2.24)，(2.25) に示す通り，SO(3) や SE(3) には満たすべき制約があるため，同様の加算を行うとこのような制約を満たさなくなります．

例えば，3 次元空間上での並進と回転を含む変化量を $\Delta T \in \mathrm{SE}(3)$ とし，以下のように定めます．

$$\Delta T = \begin{pmatrix} \Delta R & \Delta\mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tag{2.26}$$

ここで，T と ΔT の単純な加算を考えると以下になります．

$$T + \Delta T = \begin{pmatrix} R+\Delta R & \mathbf{t}+\Delta\mathbf{t} \\ \mathbf{0}^\top & 2 \end{pmatrix} \notin \mathrm{SE}(3) \tag{2.27}$$

これは明らかに式 (2.25) を満たさないため，$T+\Delta T$ は剛体変換行列にはなりません．SE(3) の制約を満たしたまま変化を反映させるためには，$T\Delta T$ を計算します[*3]．

$$T\Delta T = \begin{pmatrix} R\Delta R & R\Delta\mathbf{t}+\mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \in \mathrm{SE}(3) \tag{2.28}$$

また，$T_1, T_2 \in \mathrm{SE}(3)$ 間の差分を ΔT とした場合，$T_1\Delta T = T_2$ となるため，T_1 と T_2 の間の差分は以下のように定義されます．

$$\begin{aligned} \Delta T &= T_1^{-1} T_2 \\ &= \begin{pmatrix} R_1^\top R_2 & R_2(\mathbf{t}_2 - \mathbf{t}_1) \\ \mathbf{0}^\top & 1 \end{pmatrix} \in \mathrm{SE}(3) \end{aligned} \tag{2.29}$$

式 (2.28)，(2.29) を用いれば，SE(3)（もしくは SO(3)）の空間での状態の変化を考えることができます．しかしこれらはベクトル空間で議論される加算や減算と比較すると直感的でなく，また前節までで述べたガウス・ニュートン法が適用できる表現になっていません．そこで，リー群を用いて表せられている状態をベクトルに対応させなれないかということを考えますが，リー群に対応したベクトル空間としてリー代数を用いることができます．そして SO(3)，SE(3) に対応したリー代数をそれぞれ $\mathfrak{so}(3)$，$\mathfrak{se}(3)$ と記述します．$\mathfrak{so}(3)$ と $\mathfrak{se}(3)$ は厳密にはそれぞれ $\mathbb{R}^{3\times3}$ と $\mathbb{R}^{4\times4}$ の行列の集合ですが，それぞれ独立した 3 つと 6 つの実数で表現することができるため，3 次元，および 6 次元ベクトルとして表現できます．そして，SO(3) と $\mathfrak{so}(3)$，もしくは SE(3) と $\mathfrak{se}(3)$ 間を変換させる写像として，それぞれ**指数写像**（Exponential Map）と**対数写像**（Logarithm Map）があります．

$$\begin{aligned} &\exp : \mathfrak{so}(3) \to \mathrm{SO}(3) \\ &\log : \mathrm{SO}(3) \to \mathfrak{so}(3) \end{aligned} \tag{2.30}$$

$$\begin{aligned} &\exp : \mathfrak{se}(3) \to \mathrm{SO}(3) \\ &\log : \mathrm{SO}(3) \to \mathfrak{se}(3) \end{aligned} \tag{2.31}$$

なお，式 (2.30) と式 (2.31) に示す指数・対数写像はそれぞれ別物になります．ただし，本書で使用するにあたって適宜 SO(3) や SE(3) といった断りは入れないため，引数となっている変数でどちらの写像が使われているか判断します．

[*3] 行列の積は左右どちらから掛けるかで結果が変わるため，ΔT をどちらから作用させるかが重要になりますが，本書では基本的に動作に伴う変化に関しては右作用を用います．

リー代数を用いることで，状態量がリー群を用いて表現されている場合でも，ガウス・ニュートン法などを用いた最適化を行うことができますが，リー代数を用いた最適化を述べる前に，指数写像や対数写像について解説します．ただしこれらの写像の導出はせず，あくまで計算がどのように行われているかだけを示します．

2.5.1 指数写像と対数写像

ある 3 次元ベクトル $\boldsymbol{\phi}$ に対して，式 (2.30) に示す指数写像は以下のように定義されます．

$$\begin{aligned} \theta &= \|\boldsymbol{\phi}\|_2 \\ \exp\left(\boldsymbol{\phi}^\wedge\right) &= I_3 + \frac{\sin\theta}{\theta}\boldsymbol{\phi}^\wedge + \frac{1-\cos\theta}{\theta^2}\left(\boldsymbol{\phi}^\wedge\right)^2 \end{aligned} \tag{2.32}$$

ここで $(\cdot)^\wedge$ は，3 次元ベクトルを $\mathfrak{so}(3)$ に変換する操作（もしくは 6 次元ベクトルを $\mathfrak{se}(3)$ に変換する操作）であり，3 次元ベクトルの場合は**反対称行列**（Skew-Symmetric Matrix）を生成する操作であるともみなせます．

$$\boldsymbol{\phi}^\wedge = \begin{pmatrix} 0 & -\phi_z & \phi_y \\ \phi_z & 0 & -\phi_x \\ -\phi_y & \phi_x & 0 \end{pmatrix} \in \mathfrak{so}(3) \tag{2.33}$$

少し話が逸れますが，反対称行列の持つ性質として，$\mathbf{a}^\wedge\mathbf{b} = -\mathbf{b}^\wedge\mathbf{a}$ というものがあります（ただし $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$ です）．

$$\begin{aligned} \mathbf{a}^\wedge\mathbf{b} &= \begin{pmatrix} -a_z b_y + a_y b_z \\ a_z b_x - a_x b_z \\ -a_y b_x + a_x b_y \end{pmatrix} \\ &= \begin{pmatrix} 0 & b_z & -b_y \\ -b_z & 0 & b_x \\ b_y & -b_x & 0 \end{pmatrix} \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \\ &= -\mathbf{b}^\wedge\mathbf{a} \end{aligned} \tag{2.34}$$

この性質は，後ほどの実装でヤコビアンを計算するときに用います．

また，式 (2.30) に示す対数写像は以下のように定義されます．

$$\begin{aligned} \theta &= \arccos\left(\frac{\mathrm{tr}(R)-1}{2}\right) \\ \log(R) &= \frac{\theta}{2\sin\theta}\left(R - R^\top\right) \end{aligned} \tag{2.35}$$

なお $\log(R) = \boldsymbol{\phi}^\wedge \in \mathfrak{so}(3)$ となるため，この反対称行列から 3 次元ベクトル $\boldsymbol{\phi} = \begin{pmatrix}\phi_x & \phi_y & \phi_z\end{pmatrix}^\top$ を取り出す操作を以下のように定義します．

$$\boldsymbol{\phi} = \left(\boldsymbol{\phi}^\wedge\right)^\vee \tag{2.36}$$

後に示しますが，$\mathfrak{se}(3)$ から 6 次元ベクトルを取り出す操作も同様に $(\cdot)^\vee$ と表記します．

式 (2.32)，(2.35) では，それぞれ θ が分母に表れるため，$\theta \ll 1$ の場合計算が不安定になります．そのため $\theta \ll 1$ の場合には，以下のように近似して計算を行います．

$$\begin{aligned} \exp\left(\boldsymbol{\phi}^\wedge\right) &\simeq I_3 + \boldsymbol{\phi}^\wedge + \frac{1}{2}\left(\boldsymbol{\phi}^\wedge\right)^2 \\ \log(R) &\simeq \frac{1}{2}\left(R - R^\top\right) \end{aligned} \tag{2.37}$$

次に，SE(3) に関する指数・対数写像を考えるにあたり，$\boldsymbol{\xi} = \begin{pmatrix} \mathbf{v}^\top & \boldsymbol{\phi}^\top \end{pmatrix}^\top$ となる 6 次元ベクトルを定義します．そしてこの 6 次元ベクトルを $\mathfrak{se}(3)$ とする操作を以下のように定めます[*4]．

$$\boldsymbol{\xi}^\wedge = \begin{pmatrix} \boldsymbol{\phi}^\wedge & \mathbf{v} \\ \mathbf{0}^\top & 0 \end{pmatrix} \in \mathfrak{se}(3) \tag{2.38}$$

ここで $\boldsymbol{\phi}^\wedge$ は式 (2.33) に示す反対称行列を返す操作になります．式 (2.38) を用いて，式 (2.31) に示す指数写像は以下のように定義されます．

$$\exp\left(\boldsymbol{\xi}^\wedge\right) = \begin{pmatrix} \exp\left(\boldsymbol{\phi}^\wedge\right) & J_l\left(\boldsymbol{\phi}^\wedge\right)\mathbf{v} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tag{2.39}$$

なお $\exp\left(\boldsymbol{\phi}^\wedge\right)$ は式 (2.32) に示すものであり，$J_l\left(\cdot\right)$ は左ヤコビアンとして以下のように定義されます．

$$J_l\left(\boldsymbol{\phi}^\wedge\right) = I_3 + \frac{1-\cos\theta}{\theta^2}\boldsymbol{\phi}^\wedge + \frac{\theta-\sin\theta}{\theta^3}\left(\boldsymbol{\phi}^\wedge\right)^2 \tag{2.40}$$

ただし，$\theta = \|\boldsymbol{\phi}\|$ になります．

式 (2.31) に示す対数写像は以下のように定義されます．

$$\log\left(T\right) = \begin{pmatrix} \boldsymbol{\phi}^\wedge & J_l^{-1}\left(\boldsymbol{\phi}\right)\mathbf{t} \\ \mathbf{0}^\top & 0 \end{pmatrix} \tag{2.41}$$

ここで $J_l^{-1}\left(\cdot\right)$ は左ヤコビアンの逆行列であり，以下のように定義されます．

$$J_l^{-1}\left(\boldsymbol{\phi}\right) = I_3 - \frac{1}{2}\boldsymbol{\phi}^\wedge + \left(\frac{1}{\theta^2} - \frac{1+\cos\theta}{2\theta\sin\theta}\right)\left(\boldsymbol{\phi}^\wedge\right)^2 \tag{2.42}$$

ただし同様に $\theta = \|\boldsymbol{\phi}\|$ となります．なお $\log\left(T\right) \in \mathfrak{se}(3)$ は $\mathbb{R}^{4\times4}$ の行列であるため，ここから 6 次元のベクトル $\boldsymbol{\xi}$ を取得する操作を以下のように定めます．

$$\begin{aligned} \boldsymbol{\xi} &= \begin{pmatrix} J_l^{-1}\left(\boldsymbol{\phi}\right)\mathbf{t} \\ \boldsymbol{\phi} \end{pmatrix} \\ &= \left(\log\left(T\right)\right)^\vee \end{aligned} \tag{2.43}$$

ただし $\boldsymbol{\phi} = \left(\log\left(R\right)\right)^\vee$ となります．

なお式 (2.40)，(2.42) においても，θ が分母に含まれます．そのため，$\theta \ll 1$ となる場合は，式 (2.37) に示すように，数値計算の不安定さを回避するために近似計算を行う必要があります．

2.5.2　リー代数を介した状態更新

前述の通り，対数写像を用いることでリー群の元をリー代数の元に写像することができ，ベクトル空間上での最適化を適用することが可能となります．これにより，例えばガウス・ニュートン法のような非線形最適化アルゴリズムをリー群に対して適用することができます．ただし，最終的な状態はリー群上の元として表現されるため，リー代数を介した状態更新の方法を整理しておきます．なお以下では SE(3) を対象に議論を進めますが，SO(3) に関しても同様の考え方が適用されます．

例えばガウス・ニュートン法などを用いて，状態 $T \in \mathrm{SE}(3)$ に対するリー代数上での更新量 $\delta\boldsymbol{\xi} \in \mathbb{R}^6$ が求められたとします．この更新量は，指数写像を介してリー群の元に変換され，実際の状態更新は以下のように行われます．

$$T \leftarrow \exp\left(\delta\boldsymbol{\xi}^\wedge\right) T \tag{2.44}$$

[*4] 式 (2.33) にて $(\cdot)^\wedge$ を 3 次元ベクトルに対応する反対称行列を生成する操作としていますが，6 次元ベクトルの場合は式 (2.38) に示すように $\mathfrak{se}(3)$ を生成する操作として扱います．

この操作は，リー代数の元である $\delta\boldsymbol{\xi}$ を，リー群の元 T に加算している操作ともみなすことができ，式 (2.15) と対応させて以下のように書くこともあります．

$$T \leftarrow T \boxplus \delta\boldsymbol{\xi} \tag{2.45}$$

なお式 (2.44) では状態更新のために $\exp\left(\delta\boldsymbol{\xi}^{\wedge}\right)$ を左から掛けていますが，これは最適化計算のときの勾配を考えるときに，リー群の左からの摂動に対する残差の変化を考えているためです．もし右からの摂動を考える場合には $\exp\left(\delta\boldsymbol{\xi}^{\wedge}\right)$ を右から掛ける必要があるため注意が必要になります．

もし $\boldsymbol{\xi} = (\log(T))^{\vee}$ と表されると仮定した場合，リー代数上での新たな状態ベクトルは $\boldsymbol{\xi} + \delta\boldsymbol{\xi}$ となります．ここで $\delta\boldsymbol{\xi}$ が微小変化であるとした上で Baker-Campbell-Hausdorff（BCH）展開を考えると，以下の近似式を得ることができます．

$$\begin{aligned}\exp\left((\boldsymbol{\xi} + \delta\boldsymbol{\xi})^{\wedge}\right) &\simeq \exp(\delta\boldsymbol{\xi}^{\wedge})\exp(\boldsymbol{\xi}^{\wedge}) \\ &= \exp(\delta\boldsymbol{\xi}^{\wedge})T\end{aligned} \tag{2.46}$$

よって式 (2.44) に示すように状態更新を行うことができることが確認できます．

2.5.3 リー群を用いたヤコビアンの計算

2.3 項で述べた通り，ガウス・ニュートン法を利用するにあたってはヤコビアンを導出することが重要になります．例えば今，$T \in \mathrm{SE}(3)$ に依存する残差ベクトル $\mathbf{r} \in \mathbb{R}^N$ があるとします．このヤコビアンを求めるためには，次式を満たす J を求めれば良いことになります．

$$\mathbf{r}\left(\exp\left(\delta\boldsymbol{\xi}^{\wedge}\right)T\right) - \mathbf{r}(T) = J\delta\boldsymbol{\xi} \tag{2.47}$$

なお $J \in \mathbb{R}^{N\times 6}$（$\mathbf{r}(R) \in \mathbb{R}^N, R \in \mathrm{SO}(3)$ の場合は $\mathbb{R}^{N\times 3}$）となります．実際に残差ベクトルを定めてヤコビアンを求める例は，次章以降で述べています．

また残差ベクトルを微分するにあたり，リー群の元同士での微分を行うことがあります．これを考えるために，一例としてシンプルな $\partial T/\partial T$ について考えてみます．この場合は，以下の式を満たすヤコビアン J を求めれば良いことになります．

$$T^{-1}\left(\exp\left(\delta\boldsymbol{\xi}^{\wedge}\right)T\right) = I_4 + (J\delta\boldsymbol{\xi})^{\wedge} \tag{2.48}$$

これは式 (2.29) に示すような，$\exp\left(\delta\boldsymbol{\xi}^{\wedge}\right)T$ と T の差分を考えていることになり，特に $T^{-1}T = I_4$ であることから，リー群の恒等元周りでの微小変動を考えていることになります．そのためベクトル空間で $f(\mathbf{x}+\Delta\mathbf{x}) - f(\mathbf{x}) = J\Delta\mathbf{x}$ という式を考えていることと等価になります．ただし $J\delta\boldsymbol{\xi}$ は 6 次元のベクトルとなるため，そのままでは I_4 に加算することができないので，$(\cdot)^{\wedge}$ 作用により対応する $\mathfrak{se}(3)$ に変換していることに注意してください．そして，$T^{-1}\exp\left(\delta\boldsymbol{\xi}^{\wedge}\right)T$ は以下のように書くことができます．

$$T^{-1}\exp\left(\delta\boldsymbol{\xi}^{\wedge}\right)T = \exp\left(\left(\mathrm{Ad}_{T^{-1}}\,\delta\boldsymbol{\xi}\right)^{\wedge}\right) \tag{2.49}$$

ここで $\mathrm{Ad}_T \in \mathbb{R}^{6\times 6}$ は**随伴作用素**（Adjoint）と呼ばれます（詳細な定義は次項で述べます）．ここで $\delta\boldsymbol{\xi}$ が微小変動であることを考慮すると，以下のように近似することができます．

$$\exp\left(\left(\mathrm{Ad}_{T^{-1}}\,\delta\boldsymbol{\xi}\right)^{\wedge}\right) \simeq I_4 + \left(\mathrm{Ad}_{T^{-1}}\,\delta\boldsymbol{\xi}\right)^{\wedge} \tag{2.50}$$

式 (2.48)，(2.50) を比較すると，$J = \mathrm{Ad}_{T^{-1}}$ となることがわかり，これが $\partial T/\partial T$ の結果となります．

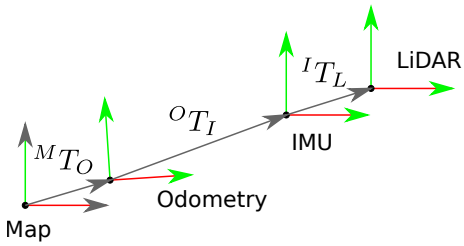

図 2.1 Coordinate frames employed in this work.

2.5.4 随伴作用素

随伴作用素とは，以下の式を満たすものとなります．

$$\begin{aligned}(\mathrm{Ad}_R\,\boldsymbol{\phi})^\wedge &= R\boldsymbol{\phi}^\wedge R^\top \\ (\mathrm{Ad}_T\,\boldsymbol{\xi})^\wedge &= T\boldsymbol{\xi}^\wedge T^{-1}\end{aligned} \tag{2.51}$$

ただし $\boldsymbol{\phi} \in \mathbb{R}^3$，$\boldsymbol{\xi} \in \mathbb{R}^6$ になります．詳細な導出方法に関しては省きますが，それぞれの随伴作用素は以下のように定義されます．

$$\mathrm{Ad}_R = R \tag{2.52}$$

$$\mathrm{Ad}_T = \begin{pmatrix} R & \mathbf{t}^\wedge R \\ 0 & R \end{pmatrix} \tag{2.53}$$

2.6 座標系とその表記

本書で扱う座標系は主に図 2.1 に示すように地図，オドメトリ，IMU，LiDAR の 4 つの座標になります．それぞれの座標は，M，O，I，L の添字により表すこととします．例えばある点 $\mathbf{p}$ がそれぞれの座標で定められる場合，左上に添え字を置きそれぞれ ${}^{M}\mathbf{p}$，${}^{O}\mathbf{p}$，${}^{I}\mathbf{p}$，${}^{L}\mathbf{p}$ と表します．またある点の座標変換（Source S から Target T への変換）を考えるとき，以下のどちらかを用いて表します．

$$\begin{aligned}{}^{T}\mathbf{p} &= {}^{T}R_S\,{}^{S}\mathbf{p} + {}^{T}\mathbf{t}_S \\ {}^{T}\mathbf{p} &= {}^{T}T_S\,{}^{S}\mathbf{p}\end{aligned} \tag{2.54}$$

ただし SE(3) を用いるときは $\mathbf{p} \in \mathbb{R}^4$ となり，4 要素目には 1 が入ることになります．

例えば，IMU 座標からオドメトリ座標に点を変換する場合は，${}^{O}\mathbf{p} = {}^{O}R_I\,{}^{I}\mathbf{p} + {}^{O}\mathbf{t}_I$，もしくは ${}^{O}\mathbf{p} = {}^{O}T_I\,{}^{I}\mathbf{p}$ と表記します．また，もし IMU 座標から地図座標への変換を直接考えるときは，以下のどちらかを用います．

$$\begin{aligned}{}^{M}\mathbf{p} &= {}^{M}R_O\left({}^{O}R_I\,{}^{I}\mathbf{p} + {}^{O}\mathbf{t}_I\right) + {}^{M}\mathbf{t}_O \\ {}^{M}\mathbf{p} &= {}^{M}T_O\,{}^{O}T_I\,{}^{I}\mathbf{p}\end{aligned} \tag{2.55}$$

ただし，${}^{M}T_I = {}^{M}T_O\,{}^{O}T_I$ とし，またこれに対応する並進ベクトルと回転行列を ${}^{M}\mathbf{t}_I$，${}^{M}R_I$ とすれば，式 (2.55) は式 (2.54) のように 1 つの変換にまとめて書くこともできます．

なお図 2.1 に示すそれぞれの座標変換ですが，基本的には，IMU と LiDAR 間の変換 ${}^{I}T_L$ は静的であるため事前にキャリブレーションで求め，LIO が ${}^{O}T_I$，SLAM（もしくは自己位置推定）が ${}^{M}T_O$ を

求めることになります．ただし，${}^{I}T_{L}$ もオンラインで最適化することもできます．また SLAM や自己位置推定を使わずに，LIO が直接地図座標への変換を求めると仮定する場合もあるため，図 2.1 に示す座標を必ず守らなければならないということではありません．

第 3 章

スキャンマッチング

3.1 スキャンマッチングの概要

スキャンマッチングとは，2 つの点群を正しく照合できるような剛体変換 $T = (R \mid \mathbf{t}) \in \mathrm{SE}(3)$ を求める処理のことです．今，点群 $\mathcal{P} = (\mathbf{p}_1, ..., \mathbf{p}_N)$ と $\mathcal{Q} = (\mathbf{q}_1, ..., \mathbf{q}_M)$ を照合させることを考えます．ただし $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$ です．このとき，以下のコスト関数を考えます[*1]．

$$E(\mathbf{t}, R) = \sum_{i=1}^{N} \|\mathbf{q}_i - (R\mathbf{p}_i + \mathbf{t})\|_2^2 \tag{3.1}$$

ここで $\mathbf{q}_i$ は，$\mathbf{p}_i$ を剛体変換した点 $R\mathbf{p}_i + \mathbf{t}$ に最も近い $\mathcal{Q}$ 内の点です．なお，このような最も近い点である最近傍点を探索するためには kd 木などを用いますが，本書で示す実装では *nanoflann*[*2]というライブラリを利用します．

式 (3.1) は，以下のように書くことも可能です．

$$E(T) = \sum_{i=1}^{N} \|\mathbf{q}_i - T\mathbf{p}_i\|_2^2 \tag{3.2}$$

なお式 (3.2) の形式で記述する場合，$\mathbf{p}, \mathbf{q} \in \mathbb{R}^4$ となり，それぞれの 4 要素目には 1 が入ることになります．そのため，$T\mathbf{p}$ も 4 要素目が 1 の 4 次元ベクトルになりますが，$\mathbf{q}$ の 4 要素目も 1 となるため，$\mathbf{q} - T\mathbf{p}$ の 4 要素目は常に 0 になります．そのため，結果としてコスト関数の値は式 (3.1) に示す値と同じになります．なお以下では，$\mathbf{q} - T\mathbf{p}$ を残差ベクトル $\mathbf{r}$ として定めます．

スキャンマッチングでは，以下に示す剛体変換を求めることを考えます．

$$T^* = \underset{T}{\operatorname{argmin}}\, E(T) \tag{3.3}$$

式 (3.3) は，コスト関数 E を最小化する姿勢 T^* を求めるという意味になります．この T^* は，一般的に反復処理を行うことで求めるられるため，Iterative Closest Points（ICP）スキャンマッチングとも呼ばれます．なお式 (3.2) に示すコストを最小にするスキャンマッチングは，対応する点同士の距離を最小にするため，point-to-point ICP とも呼ばれます．

point-to-point ICP は一般的にノイズに脆弱であるといわれています．そのため本書では，より頑健性の高い点と面の距離を最小化する point-to-plane ICP について考えます．point-to-plane ICP では，

[*1] 実装では式 (2.17) に示すフーバー損失を考えますが，本章では説明が煩雑になることを防ぐためにフーバー損失は無視します

[*2] `https://github.com/jlblancoc/nanoflann`

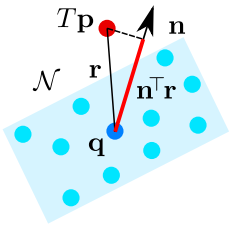

図 3.1　Residual error used in point-to-plane ICP.

以下のコスト関数を考えます．

$$E(T) = \sum_{i=1}^{N} \left(\mathbf{n}_i^\top \mathbf{r}_i\right)^2 \tag{3.4}$$

ここで $\mathbf{n}_i$ は，$\mathbf{q}_i$ の周辺の点を用いて計算した面の 3 次元空間内での法線ベクトルであり，$r = \mathbf{n}_i^\top \mathbf{r}_i$ を残差として考えます．図 3.1 にこれらの関係をまとめます．なお，$\mathbf{r}$ が 4 次元ベクトルであるため $\mathbf{n}$ も 4 次元ベクトルとなりますが，$\mathbf{r}$ の 4 要素目は常に 0 であるため，$\mathbf{n}$ の 4 要素目がいくつであっても計算に違いは表れません．

以下ではまず，法線ベクトルの計算方法を述べた後に，残差ベクトルに関するヤコビアンを導出し，このヤコビアンを用いてガウス・ニュートン法により最適化を行う流れについて解説していきます．

3.2　法線ベクトルの計算

ある点 $\mathbf{q}$ に対する法線ベクトルを計算するにあたり，まず $\mathbf{q}$ 周辺の点を取得し，この集合を $\mathcal{N}$ とします．そしてこれを用いて，まず平均 $\bar{\mathbf{q}}$ を計算します．

$$\bar{\mathbf{q}} = \frac{1}{|\mathcal{N}|} \sum_{\mathbf{q}_i \in \mathcal{N}} \mathbf{q}_i \tag{3.5}$$

次に $\mathcal{N}$ の共分散行列 C を求めます．

$$C = \frac{1}{|\mathcal{N}|} \sum_{\mathbf{q}_i \in \mathcal{N}} \left(\mathbf{q}_i - \bar{\mathbf{q}}\right)\left(\mathbf{q}_i - \bar{\mathbf{q}}\right)^\top \tag{3.6}$$

そして C を固有分解します．

$$C = V \Lambda V^\top \tag{3.7}$$

ここで $V = (\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3)$，$\Lambda = \mathrm{diag}\,(\lambda_1, \lambda_2, \lambda_3)$ であり，$\mathbf{v}_i$ と λ_i は固有ベクトルとそれに対応する固有値になります．

固有値は，各方向における点のばらつきを表す量であり，最小の固有値に対応する固有ベクトルは，点群が最もばらついていない方向，すなわち最も潰れている方向を表します．そのため，この最小固有値 $\lambda_{\min}$ が十分に小さい場合，対応する固有ベクトル $\mathbf{v}_{\min}$ は，点群が存在する局所平面の法線ベクトルとしてみなすことができます．このようにして得られた $\mathbf{v}_{\min}$ を，点 $\mathbf{q}$ における法線ベクトル $\mathbf{n}$ として定義します．なお，$\lambda_{\min}$ に対して閾値を設定することで，平面性が不十分な点を除外することも可能です．

3.3 ヤコビアンの計算

point-to-plane の ICP をガウス・ニュートン法を用いて解くにあたり，残差 $r = \mathbf{n}^\top \mathbf{r}$ の姿勢 T に関するヤコビアンを求めます．このヤコビアンは，連鎖則を用いて以下のように計算できます．

$$\frac{\partial r}{\partial T} = \frac{\partial r}{\partial \mathbf{r}} \frac{\partial \mathbf{r}}{\partial T} \tag{3.8}$$

ここで，$r = \mathbf{n}^\top \mathbf{r}$ であるため，$\partial r / \partial \mathbf{r}$ は明らかに $\mathbf{n}^\top$ です．そのため以下では，$\partial \mathbf{r} / \partial T$ についてのみ詳細の計算方法を示します．

残差ベクトル $\mathbf{r}$ は，明らかに姿勢 T に関する関数になっています．そのため，次式を考えることでヤコビアン J を導出します．

$$\mathbf{r}\left(\exp\left(\delta\boldsymbol{\xi}^\wedge\right) T\right) - \mathbf{r}\left(T\right) = J\delta\boldsymbol{\xi} \tag{3.9}$$

なお $\delta\boldsymbol{\xi}^\wedge = \begin{pmatrix} \delta\mathbf{v}^\top & \delta\boldsymbol{\phi}^\top \end{pmatrix}^\wedge \in \mathfrak{se}(3)$ であり，$\exp\left(\delta\boldsymbol{\xi}^\wedge\right) T$ は SE(3) の空間で T を左から微小摂動させた状態となります．式 (3.9) の左辺を展開していくと以下が得られます．

$$\begin{aligned}
\mathbf{q} - \exp\left(\delta\boldsymbol{\xi}^\wedge\right) T\mathbf{p} - \left(\mathbf{q} - T\mathbf{p}\right) &= -\left(\exp\left(\delta\boldsymbol{\xi}^\wedge\right) - I_4\right) T\mathbf{p} \\
&= -\left(\begin{pmatrix} I_3 + \delta\boldsymbol{\phi}^\wedge & \delta\mathbf{v} \\ \mathbf{0}^\top & 1 \end{pmatrix} - I_4\right) \begin{pmatrix} R\mathbf{p} + \mathbf{t} \\ 1 \end{pmatrix} \\
&= -\begin{pmatrix} \delta\boldsymbol{\phi}^\wedge & \delta\mathbf{v} \\ \mathbf{0}^\top & 0 \end{pmatrix} \begin{pmatrix} \mathbf{p}' \\ 1 \end{pmatrix} \\
&= -\begin{pmatrix} \delta\boldsymbol{\phi}^\wedge \mathbf{p}' + \delta\mathbf{v} \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \left(\mathbf{p}'\right)^\wedge \delta\boldsymbol{\phi} - \delta\mathbf{v} \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} -I_3 & \left(\mathbf{p}'\right)^\wedge \\ \mathbf{0}^\top & \mathbf{0}^\top \end{pmatrix} \delta\boldsymbol{\xi}
\end{aligned} \tag{3.10}$$

なお $\mathbf{p}' = R\mathbf{p} + \mathbf{t}$ と置き，$\delta\boldsymbol{\phi}^\wedge \mathbf{p}' = -\left(\mathbf{p}'\right)^\wedge \delta\boldsymbol{\phi}$ を用いました．よって，残差に対するヤコビアンは以下となります．

$$\begin{aligned}
\frac{\partial r}{\partial T} =& \mathbf{n}^\top \begin{pmatrix} -I_3 & \left(\mathbf{p}'\right)^\wedge \\ \mathbf{0}^\top & \mathbf{0}^\top \end{pmatrix} \\
=& \begin{pmatrix} -\mathbf{n}^\top & \mathbf{n}^\top \left(\mathbf{p}'\right)^\wedge \end{pmatrix} \in \mathbb{R}^{1\times 6}
\end{aligned} \tag{3.11}$$

ただし最後の $\mathbf{n} \in \mathbb{R}^3$ は 3 次元の法線ベクトルになります．

3.4　ガウス・ニュートン法を用いた最適化

式 (3.11) に示すヤコビアンを用いて，point-to-plane の ICP スキャンマッチングをガウス・ニュートン法で解くにあたり，ヘッセ行列と勾配ベクトルを以下のように求めます．

$$\begin{aligned} H &= \sum_{i=1}^{N} J_i^\top J_i \\ \mathbf{b} &= \sum_{i=1}^{N} J_i^\top r_i \end{aligned} \tag{3.12}$$

ただしフーバー損失を用いる場合は，式 (2.22) を用いて $w_i = \rho'\left(r_i^2\right)$ を定め，$H = \sum_i w_i J_i^\top J_i$，$\mathbf{b} = \sum_i w_i J_i^\top r_i$ を計算します．そして，式 (3.12) に示す H と $\mathbf{b}$ を用いて，$H\delta\boldsymbol{\xi} = -\mathbf{b}$ を満たす $\delta\boldsymbol{\xi}$ を求めます．最後に，以下のように状態を更新します．

$$T \leftarrow \exp\left(\delta\boldsymbol{\xi}^\wedge\right) T \tag{3.13}$$

この更新を，$\|\delta\boldsymbol{\xi}\|_2 \leq \epsilon$，もしくは規定回数繰り返すまで実行することで，式 (3.3) に示す T^* が得られたとみなします．なお ϵ は任意の定数です．

3.5　実用にあたって

スキャンマッチングを正確に実行するためには，いかに良い初期値を得るかが極めて重要になります．初期値の精度が高ければ，対応点探索は安定して動作し，スキャンマッチングは高い精度で最適解に収束します．一方で初期値の誤差が大きくなると対応点探索に失敗しやすくなり，スキャンマッチングはうまく機能しなくなります．なお，スキャンマッチングを実行した結果，コスト関数を最小にしない解に収束してしまうこともありますが，このような解は**局所最適解**（Local Minima）と呼ばれます．

このような誤対応の影響を軽減するために，ロバストカーネルであるフーバー損失を導入することが有効です．フーバー損失は，大きな残差の影響を抑えることで，誤対応による最適化の劣化をある程度防ぐことができます．しかし，フーバー損失の導入は万能ではありません．例えば，多数の誤対応の中に少数の正しい対応が含まれている場合でも，それらの残差が大きければ，フーバー損失により重みが下がり，正しい対応の影響までもが過小評価されてしまうことがあります．その結果，フーバー損失を使わない場合にスキャンマッチングが成功し，逆に導入したことで失敗するケースもあり得ます．したがって，フーバー損失のパラメータ δ は慎重に調整する必要があります．それでも一般的には，フーバー損失を導入することでスキャンマッチングのロバスト性は向上します．

なお，スキャンマッチング単体では良好な初期値を毎フレーム安定して得ることは困難です．特に移動速度や回転速度が大きい場合，その影響は顕著になります．このような問題を補うためには，IMU などの外部センサと併用して，より正確な初期位置を与えることが効果的です．次章では，LiDAR と IMU をルーズカップリングにより統合する手法について解説します．

また初期値だけでなく，スキャンマッチングを行う環境にも注意しなければなりません．スキャンマッチングは基本的に，幾何的な拘束を基に剛体変換を求めるため，正しい解が得られるためには良い拘束が得られる環境である必要があります．もし適切な拘束が得られない場合にガウス・ニュートン法を解くと，$H\delta\boldsymbol{\xi} = -\mathbf{b}$ を満たす $\delta\boldsymbol{\xi}$ を適切に求められなくなります．これは H が正則でなくなり，この逆行列が定まらなくなるためです．このような例は**縮退**（Degeneracy）したケースと呼ばれ，スキャ

ンマッチングがそもそも機能しないケースとなります．なお縮退を検出する方法として，式 (3.12) に示すヘッセ行列の固有値分解を行い，その最小固有値を確認する方法もあります．ヘッセ行列の最小固有値が極端に小さい場合，行列がランク落ちして正しい逆行列が求められなくなり，結果として最適化が破綻することになります．

第 4 章

ルーズカップリングに基づく LIO

4.1 状態量と問題設定

前章で述べたスキャンマッチングでは，姿勢 $T \in \mathrm{SE}(3)$ のみを求める問題を考えていました．これに対して本章で述べる LIO では，以下の状態量を求めることを考えます．

$$\mathbf{x} = \left({}^{O}\mathbf{t}_I \ {}^{O}R_I \ {}^{O}\mathbf{v} \ \mathbf{b}^{\omega} \ \mathbf{b}^{a}\right) \tag{4.1}$$

ここで ${}^{O}\mathbf{t}_I \in \mathbb{R}^3$ と ${}^{O}R_I \in \mathrm{SO}(3)$ は，オドメトリ座標での IMU の姿勢を表す並進ベクトルと回転行列，${}^{O}\mathbf{v} \in \mathbb{R}^3$ はオドメトリ座標系における IMU の速度ベクトル，$\mathbf{b}^{\omega}, \mathbf{b}^{a} \in \mathbb{R}^3$ は IMU の角速度と加速度に対するバイアスになります．なお，$(\log(R))^{\vee} \in \mathbb{R}^3$ となるので，本章で述べる LIO では 15 次元の状態を推定する問題となります．また，IMU の姿勢を求めていることに注意してください．

LIO では LiDAR と IMU を用いるため，各センサデータの時間軸を考えることが重要になります．例えば LiDAR が時刻 $t-1$ および t において，それぞれ $\mathcal{P}_{t-1}$，$\mathcal{P}_t$ の点群を取得するとします．この間，IMU は $\mathcal{U}_t = \left(\boldsymbol{\omega}_t^1 \ \mathbf{a}_t^1 \ \cdots \ \boldsymbol{\omega}_t^{M_t} \ \mathbf{a}_t^{M_t}\right)$ のデータを計測するとします[*1]．なお図 4.1 に，これらのセンサデータ，および本章で使われる時間の関係を図示しています．

本章で解説する LIO では，LiDAR と IMU の計測値を用いて，式 (4.1) に示す状態量を求めることが目標になり，これは以下の処理を繰り返すことで達成されます．

1. IMU プレインテグレーションを用いた状態の予測
2. LiDAR 点群の歪み補正
3. 局所地図と LiDAR 点群のスキャンマッチング
4. 予測およびスキャンマッチングの結果に基づく状態の再更新
5. キーフレームの検出と局所地図の構築

なお 4 の再更新の部分がルーズカップリングに相当し，これには DirectLIO [**?**] で用いられている Hierarchical Geometric Observer（HGO）[**?**] を用います．ただし HGO の利用がルーズカップリングとして良い方法であるということではなく，実装が用意なので紹介しているということに留意ください．実装できるなら，拡張カルマンフィルたのような実装をするほうが良いといえます．

[*1] 一般に IMU の方が LiDAR より計測周期が高いため，LiDAR が計測を行う間に IMU の計測値は複数個存在します．

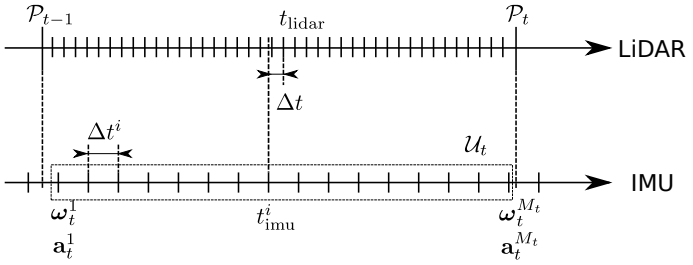

図 4.1　Temporal relationship between LiDAR and IMU measurements.

4.2　IMU プレインテグレーション

今時刻 $t-1$ の状態 $\mathbf{x}_{t-1}$ まで推定が行われていて，時刻 t の IMU のデータ $\mathcal{U}_t$ が得られているとします．スキャンマッチングを実行するにあたり，**IMU プレインテグレーション**（IMU Preitegration）を用いて，回転行列，並進ベクトル，および速度ベクトルを以下のように更新します．

$$
\begin{aligned}
{}^O\mathbf{t}_{I,t} &= {}^O\mathbf{t}_{I,t-1} + \sum_{i=1}^{M_t} {}^O\mathbf{v}_{t-1}^j \Delta t^i + \frac{1}{2} {}^O R_{I,t-1}^j \left(\mathbf{a}_t^i - \mathbf{b}_{t-1}^a\right) \left(\Delta t^i\right)^2 + \frac{1}{2}\mathbf{g}\left(\Delta t^i\right)^2 \\
{}^O R_{I,t} &= {}^O R_{I,t-1} \prod_{i=1}^{M_t} \exp\left(\left(\boldsymbol{\omega}_t^i - \mathbf{b}_{t-1}^\omega\right)^\wedge \Delta t^i\right) \\
{}^O\mathbf{v}_t &= {}^O\mathbf{v}_{t-1} + \sum_{i=1}^{M_t} {}^O R_{I,t-1}^j \left(\mathbf{a}_t^i - \mathbf{b}_{t-1}^a\right) \Delta t^i + \mathbf{g}\Delta t^i
\end{aligned}
\tag{4.2}
$$

ここで $\mathbf{g}$ は重力加速度ベクトル，Δt^i は i 番目から $i-1$ 番目の IMU の計測時間の差，${}^O R_{I,t-1}^j$ と ${}^O\mathbf{v}_{I,t-1}^j$ はそれぞれ以下となります．

$$
\begin{aligned}
{}^O R_{I,t-1}^j &= {}^O R_{I,t-1} \prod_{i=1}^{j} \exp\left(\left(\boldsymbol{\omega}_t^i - \mathbf{b}_{t-1}^\omega\right)^\wedge \Delta t^i\right) \\
{}^O\mathbf{v}_{t-1}^j &= {}^O\mathbf{v}_{t-1} + \sum_{i=1}^{j} {}^O R_{I,t-1}^j \left(\mathbf{a}_t^i - \mathbf{b}_{t-1}^a\right) \Delta t^i + \mathbf{g}\Delta t^i
\end{aligned}
\tag{4.3}
$$

次に示すスキャンマッチングでは，式 (4.2) に示す並進ベクトルと回転行列を初期値として用います．

4.3　歪み補正

一般的に IMU の計測周期は LiDAR の計測周期より高いです．そのため，LiDAR が 1 周期分の点群を計測する間に，複数の IMU の計測値を取得することが可能です．そしてこの計測値を利用すると，式 (4.2) に示すように，LiDAR が 1 周期分の点群を取得する間の姿勢を計算することができます．そしてこれらの姿勢を利用すると，LiDAR が計測した各点をより正確な位置に補正することができます．この操作を，LiDAR のデータの歪み補正と呼びます．

歪み補正の解説を行う前に，まず IMU プレインテグレーションにより得られた M_t+1 個の姿勢の集合を $\left({}^O T_I^0, \cdots, {}^O T_I^{M_t}\right)$ とし，またこれらの姿勢列に対応した M_t+1 個の時間の集合を $\left(t_{\text{imu}}^0, \cdots, t_{\text{imu}}^{M_t}\right)$ とします．なお，${}^O T_I^j = \left({}^O R_{I,t-1}^j \mid {}^O\mathbf{t}_{I,t-1}^j\right) \in \mathrm{SE}(3)$ です．また，${}^O T_I^0 = {}^O T_{I,t-1}$，${}^O T_I^{M_t} = {}^O T_{I,t}$ となります．

歪み補正を行う前提として，LiDAR が計測した各点にタイムスタンプが付与されていることを前提とします．そしてこのタイムスタンプを用いて，$t_{\mathrm{imu}}^{i} \leq t_{\mathrm{lidar}} \leq t_{\mathrm{imu}}^{i+1}\ (i = 0, \cdots, M_t - 1)$ となる IMU の計測値を探索します．今，i 番目と $i+1$ 番目の時間の間で LiDAR が点 ${}^{L}\mathbf{p}$ を計測していたとします．このとき，この点を計測した際の IMU の姿勢を以下のように求めます．

$$\begin{aligned} \Delta t &= t_{\mathrm{lidar}} - t_{\mathrm{imu}}^{i} \\ {}^{O}R_{I}^{d} &= {}^{O}R_{I,t-1}^{i} \exp\left(\left(\boldsymbol{\omega}_{t}^{i} - \mathbf{b}_{t-1}^{\omega}\right)^{\wedge} \Delta t\right) \\ {}^{O}\mathbf{t}_{I}^{d} &= {}^{O}\mathbf{t}_{I,t-1}^{i} + {}^{O}\mathbf{v}_{t-1}^{i}\Delta t + \frac{1}{2}{}^{O}R_{I,t-1}^{i}\left(\mathbf{a}_{t}^{i} - \mathbf{b}_{t-1}^{a}\right)\Delta t^{2} + \frac{1}{2}\mathbf{g}\Delta t^{2} \end{aligned} \tag{4.4}$$

そして，${}^{O}T_{I}^{d} = \left({}^{O}R_{I}^{d} \mid {}^{O}\mathbf{t}_{I}^{d}\right)$ を定め，${}^{L}\mathbf{p}$ を以下のように IMU 座標の点に変換します．

$${}^{I}\mathbf{p} = \left({}^{O}T_{I}^{M_t}\right)^{-1} {}^{O}T_{I}^{dI}T_{L}\,{}^{L}\mathbf{p} \tag{4.5}$$

ここで ${}^{I}T_{L}$ は LiDAR と IMU 間の剛体変換を表す行列であり，事前に求められているものとしています．この操作をすべての計測点群に対して行うことで，LiDAR が計測した点群の歪みを補正することができます．なお一番最後に $\left({}^{O}T_{I}^{M_t}\right)^{-1}$ を適用することで，式 (4.2) で予測した IMU の姿勢を原点とした座標の点群が得られます．

4.4 局所地図とのスキャンマッチング

IMU プレインテグレーションによる予測，および歪み補正を終えた後に，**局所地図**（Local Map）とのスキャンマッチングを実施します．なお説明の都合上先に局所地図とのスキャンマッチングについて述べますが，局所地図の作成方法に関しては 4.6 節で述べます．またスキャンマッチングに関しては基本的に 3 章で述べた方法を用いますが，本節では使用されるデータや座標に関して整理しておきます．

まず局所地図を表す点群 ${}^{O}\mathcal{M}$ が，オドメトリ座標上で構築されているとします[*2]．また前節で述べた歪み補正も適用され，LiDAR の計測点群は IMU 座標での点群 ${}^{I}\mathcal{P}$ が得られているとします．このとき，LiDAR の計測点 ${}^{I}\mathbf{p}$ に対する残差ベクトルを以下のように定めます．

$$\mathbf{r} = {}^{O}\mathbf{q} - {}^{O}T_{I}\,{}^{I}\mathbf{p} \tag{4.6}$$

なお ${}^{O}\mathbf{q}$ は，${}^{O}\mathcal{M}$ の点で ${}^{O}T_{I}\,{}^{I}\mathbf{p}$ に最も近い点です．そして，以下に示すコスト関数の最小化を行います．

$$E\left({}^{O}T_{I}\right) = \sum_{i=1}^{N} \rho\left(\left\|\mathbf{n}_{i}^{\top}\mathbf{r}_{i}\right\|_{2}^{2}\right) \tag{4.7}$$

ただし $\mathbf{n}$ は，${}^{O}\mathbf{q}$ に対応する法線ベクトルになります．

4.5 ルーズカップリングによる状態量の更新

IMU プレインテグレーションにより予測された姿勢を ${}^{O}\hat{T}_{I}$，スキャンマッチングにより得られた姿勢を ${}^{O}T_{I}^{*}$ とします．またこれらに対応する並進ベクトルと回転行列に対応するクォータニオンをそれぞ

[*2] 局所座標の構築方法にも様々な方法があり，明示的にオドメトリ座標で地図構築を行わない方法もあります．

れ ${}^{O}\hat{\mathbf{t}}_I$，${}^{O}\mathbf{t}_I^*$，${}^{O}\hat{\mathbf{q}}_I$，${}^{O}\mathbf{q}_I^*$ とします．これらを用いて，最新の状態をそれぞれ以下のように更新します．

$$\begin{aligned}
{}^{O}\mathbf{q}_I &= {}^{O}\hat{\mathbf{q}}_I + \Delta t \gamma_1 {}^{O}\hat{\mathbf{q}}_I \begin{pmatrix} 1 - \left|q_w^d\right| \\ \operatorname{sgn}\left(q_w^d\right)\mathbf{q}_v^d \end{pmatrix} \\
\mathbf{b}^{\omega} &= \hat{\mathbf{b}}^{\omega} - \Delta t \gamma_2 q_w^d \mathbf{q}_v^d \\
{}^{O}\mathbf{t}_I &= {}^{O}\hat{\mathbf{t}}_I + \Delta t \gamma_3 \mathbf{t}^d \\
{}^{O}\mathbf{v} &= {}^{O}\hat{\mathbf{v}}_t + \Delta t \gamma_4 \mathbf{t}^d \\
\mathbf{b}^{a} &= \hat{\mathbf{b}}^{a} - \Delta t \gamma_5 {}^{O}\hat{R}_I^{\top}\mathbf{t}^d
\end{aligned} \tag{4.8}$$

ここで Δt は IMU プレインテグレーションを行った時間の総和，γ_{1-5} は任意の正の定数，$\mathbf{q}^d = {}^{O}\hat{\mathbf{q}}_I^{-1} \otimes {}^{O}\mathbf{q}_I^*$，$\mathbf{t}^d = {}^{O}\mathbf{t}_I^* - {}^{O}\hat{\mathbf{t}}_I$ です．また $\mathbf{q}^d = \left(q_w^d \ \left(\mathbf{q}_v^d\right)^{\top}\right)^{\top}$，$\mathbf{q}_v^d = \begin{pmatrix} q_x^d & q_y^d & q_z^d \end{pmatrix}^{\top}$ であり，${}^{O}\hat{R}_I$ は ${}^{O}\hat{\mathbf{q}}_I$ に対応する回転行列です．

4.6　局所地図の構築

局所地図を構築する方法も様々ありますが，本書ではキーフレームを用いた方法を採用します．具体的には，LIO が推定する姿勢の中からいくつかの姿勢をキーフレームとして選択し，その姿勢とそれに対応する LiDAR の点群を用いて局所地図 ${}^{O}\mathcal{M}$ の作成を行います．キーフレームの検出にあたっては，最も単純な方法ではありますが，移動量に対して閾値を設け，前回検出したキーフレームからの並進移動量，もしくは回転量が一定値を超えた場合に，新たにキーフレームとして検出を行います．

今キーフレームの集合として $\left({}^{O}T_{I,1}, \cdots, {}^{O}T_{I,K}\right)$，またこれらのキーフレームに対応した LiDAR の点群 $\left({}^{I}\mathcal{P}_1, \cdots, {}^{I}\mathcal{P}_K\right)$ があるとします．なお K は局所地図を構築するために使用するキーフレームの数です．これらの点すべてを対応するキーフレームでオドメトリ座標に変換した点の集合を局所地図として定めます．

$${}^{O}\mathcal{M} = \bigcup_{i=1}^{K} \bigcup_{{}^{I}\mathbf{p} \in {}^{I}\mathcal{P}_i} {}^{O}T_{I,i}{}^{I}\mathbf{p} \tag{4.9}$$

キーフレームをいくつ利用するかにもよりますが，通常局所地図は大きな点群となります．そのため，すべての点に対して法線ベクトルを計算すると非常に大きな計算コストが発生します．また局所地図が大きくなると，スキャンマッチングに使用されない点も多く含まれてきます．そのため本実装では，式 (4.6) に示す残差ベクトルが定義されたときに，その点に対応する法線ベクトルの計算を行うことにします．また，計算済みかどうかを判定するフラグも実装し，局所地図構築に関する計算コストの削減を行っています．

4.7　実用にあたって

ルーズカップリングに基づく LIO は，次章で述べるタイトカップリングに基づく LIO と比較すると，実装やパラメータ調整が容易に行うことができます．加えて，おおよその環境では十分な性能を持って機能します．そのため，まず LIO を実装し，LiDAR と IMU を融合させる方法を知りたいという方には，適した手法であるといえます．ただし多くの場合，タイトカップリングに基づく LIO の方が精度が高い傾向にあります．ただしタイトカップリングに基づく LIO は実装やパラメータ調整の難易度が上がります．繰り返しにはなりますが，ルーズカップリングに基づく LIO も十分な性能を持つため，ルーズカップリングが良いかタイトカップリングが良いかは，ユーザーの状況次第になるといえます．

LIO で最も計算コストがかかる部分は，局所地図の構築部分になります．局所地図はサイズを小さくすれば構築の計算コストは下がりますが，その分スキャンマッチングに利用できる範囲が限定されるため，移動量推定においてドリフト誤差が発生しやすくなってしまいます．しかし局所地図のサイズを大きくしすぎると，地図構築の計算コストが増大し，最悪の場合，LiDAR の計測周期を超えるような計算時間となり，LIO の破綻に繋がってしまいます．

また局所地図の更新頻度も LIO の精度に大きく関わります．特に LiDAR の観測がスキャンマッチングに適さないような環境では，できる限り局所地図を更新する頻度を向上させた方が LIO の精度を維持することができます．しかし当然ながら，局所地図の更新頻度が増えるほど計算コストの増大にも繋がります．また本実装では，単純な移動量に対して閾値を設けて局所地図の更新をおこなっているため，局所地図の更新頻度を高くすると，局所地図として地図化できる範囲が小さくなり，これもドリフト誤差を発生させやすくなる要因になります．LIO の精度が低下する場合などは，局所地図に関するパラメータの調整，またもしくは，局所地図の更新ルールを見直すことが効果的なことが多いです．なおこれらの局所地図に関する問題は，次章で述べるタイトカップリングに基づく LIO でも同様に表れます．

第 5 章

タイトカップリングに基づく LIO

5.1　状態量と問題設定

前章では，推定対象の状態として IMU の姿勢（並進ベクトル ${}^{O}\mathbf{t}_{I}$ と回転行列 ${}^{O}R_{I}$），速度 ${}^{O}\mathbf{v}$，および IMU の角速度と加速度に対する計測バイアス $\mathbf{b}^{\omega}$，$\mathbf{b}^{a}$ としていました．タイトカップリングを用いた LIO も同様の状態で実装可能ですが，本章では拡張性を考慮し，状態量を以下とします．

$$\mathbf{x} = \left({}^{O}\mathbf{t}_{I}\ {}^{O}R_{I}\ {}^{O}\mathbf{v}\ \mathbf{b}^{\omega}\ \mathbf{b}^{a}\ \mathbf{g}\ {}^{I}\mathbf{t}_{L}\ {}^{I}R_{L}\right) \tag{5.1}$$

ここで $\mathbf{g} \in \mathbb{R}^3$ は重力加速度ベクトル，${}^{I}\mathbf{t}_{L} \in \mathbb{R}^3$ と ${}^{I}R_{L} \in \mathrm{SO}(3)$ は LiDAR と IMU 間の相対姿勢を表す並進ベクトルと回転行列です．なお，$\left(\log\left({}^{I}R_{L}\right)\right)^{\vee} \in \mathbb{R}^3$ となるので，本章で扱う LIO では 24 次元の状態推定を行います．また前章で述べた LIO では扱いませんでしたが，推定状態に対する共分散行列 $\Sigma \in \mathbb{R}^{24\times 24}$ も扱います．

5.2　IMU プレインテグレーションによる更新

本章で述べる LIO でも，4.2 節で述べた IMU プレインテグレーションを用いて，IMU の姿勢と速度の更新を行います．ただし本章で述べる LIO では，これらに加えて共分散行列の更新も行います．

共分散行列の更新を考えるにあたり，まずホワイトノイズベクトル $\boldsymbol{\eta} = \left(\boldsymbol{\eta}^{\omega}\ \boldsymbol{\eta}^{a}\ \boldsymbol{\eta}^{b^{\omega}}\ \boldsymbol{\eta}^{b^{a}}\right)^{\top} \in \mathbb{R}^{12}$ を導入します．なお $\boldsymbol{\eta}^{\omega}, \boldsymbol{\eta}^{a}, \boldsymbol{\eta}^{b^{\omega}}, \boldsymbol{\eta}^{b^{a}} \in \mathbb{R}^3$ はそれぞれ IMU の角速度と加速度の計測値，およびそれらの計測バイアスに加わるホワイトノイズとします．今，$\mathbf{u}_t = \left(\boldsymbol{\omega}_t^{\top}\ \mathbf{a}_t^{\top}\right)^{\top} \in \mathbb{R}^6$ として，1 つの IMU の計測値に対する更新則を考えます．これらの条件を用いると，IMU プレインテグレーションに基づく状態の更新は以下のように定めることができます．

$$\begin{aligned}
{}^{O}\mathbf{t}_{I,t} &= {}^{O}\mathbf{t}_{I,t-1} + {}^{O}\mathbf{v}_{t-1}\Delta t + \frac{1}{2}{}^{O}R_{I,t-1}\left(\mathbf{a}_t - \mathbf{b}^{a}_{t-1} - \boldsymbol{\eta}^{a}_{t}\right)\Delta t^2 + \frac{1}{2}\mathbf{g}\Delta t^2 \\
{}^{O}R_{I,t} &= {}^{O}R_{I,t-1}\exp\left(\left(\boldsymbol{\omega}_t - \mathbf{b}^{\omega}_{t-1} - \boldsymbol{\eta}^{\omega}_{t}\right)^{\wedge}\Delta t\right) \\
{}^{O}\mathbf{v}_{t} &= {}^{O}\mathbf{v}_{t-1} + {}^{O}R_{I,t-1}\left(\mathbf{a}_t - \mathbf{b}^{a}_{t-1} - \boldsymbol{\eta}^{a}_{t}\right)\Delta t + \mathbf{g}\Delta t \\
\mathbf{b}^{\omega}_{t} &= \mathbf{b}^{\omega}_{t-1} + \boldsymbol{\eta}^{b^{\omega}}_{t}\Delta t \\
\mathbf{b}^{\omega}_{t} &= \mathbf{b}^{a}_{t-1} + \boldsymbol{\eta}^{b^{a}}_{t}\Delta t \\
\mathbf{g}_{t} &= \mathbf{g}_{t-1} \\
{}^{I}\mathbf{t}_{L,t} &= {}^{I}\mathbf{t}_{L,t-1} \\
{}^{I}R_{L,t} &= {}^{I}R_{L,t-1}
\end{aligned} \tag{5.2}$$

ここで Δt は，IMU の計測にかかった時間です．

式 (5.2) による状態の更新を $\mathbf{x}_t = \mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{u}_t, \boldsymbol{\eta}_t\right)$ と記述することとします．このとき，共分散行列の更新は以下のようにすることで行うことができます．

$$\Sigma_t = F_x \Sigma_{t-1} F_x^\top + F_\eta Q F_\eta^\top \tag{5.3}$$

ここで，$F_x = \partial \mathbf{f} / \partial \mathbf{x}_{t-1} \in \mathbb{R}^{24\times 24}$，$F_\eta = \partial \mathbf{f} / \partial \boldsymbol{\eta}_t \in \mathbb{R}^{24\times 12}$ となり，$Q \in \mathbb{R}^{12\times 12}$ はプロセスノイズ共分散行列です．これらは大きな行列なので計算は煩雑になりますが，それぞれ以下のように求めることができます[*1]．

$$F_x = \begin{pmatrix} I_3 & -A\Delta t^2 & I_3\Delta t & 0 & -\frac{1}{2}{}^O R_{I,t-1}\Delta t^2 & \frac{1}{2} I_3 \Delta t & 0 & 0 \\ 0 & J_l^{-1}\left({}^O\mathbf{r}_I\right){}^O R_{I,t}^\top & 0 & -J_l^{-1}\left({}^O\mathbf{r}_I\right) J_r\left(\Delta\boldsymbol{\phi}_t\right)\Delta t & 0 & 0 & 0 & 0 \\ 0 & -A\Delta t & I_3 & 0 & -{}^O R_{I,t-1}\Delta t & I_3\Delta t & 0 & 0 \\ 0 & 0 & 0 & I_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & J_l^{-1}\left({}^I\mathbf{r}_L\right) \end{pmatrix} \tag{5.4}$$

$$F_\eta = \begin{pmatrix} 0 & -\frac{1}{2}{}^O R_{I,t-1}\Delta t^2 & 0 & 0 \\ -J_l^{-1}\left({}^O\mathbf{r}_I\right) J_r\left(\Delta\boldsymbol{\phi}_t\right)\Delta t & 0 & 0 & 0 \\ 0 & {}^O R_{I,t-1}\Delta t & 0 & 0 \\ 0 & 0 & I_3\Delta t & 0 \\ 0 & 0 & 0 & I_3\Delta t \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{5.5}$$

なお，$\hat{\boldsymbol{\omega}}_t = \boldsymbol{\omega}_t - \mathbf{b}_{t-1}^\omega - \boldsymbol{\eta}_t^\omega$，$\hat{\mathbf{a}}_t = \mathbf{a}_t - \mathbf{b}_{t-1}^a - \boldsymbol{\eta}_t^a$，$A = \left({}^O R_{I,t-1}\hat{\mathbf{a}}_t\right)^\wedge$，${}^O\mathbf{r}_I = \left(\log\left({}^O R_{I,t}\right)\right)^\vee$，${}^I\mathbf{r}_L = \left(\log\left({}^I R_{L,t}\right)\right)^\vee$，$\Delta\boldsymbol{\phi}_t = \hat{\boldsymbol{\omega}}_t\Delta t$ として表記を短縮しています．また式 (5.4)，(5.5) に示す 0 はすべて $\mathbb{R}^{3\times 3}$ の要素がすべて 0 の行列です．

5.3　IEKF による更新

IMU プレインテグレーションによる更新を終えた後に，4.3 節で述べた LiDAR 点群の歪み補正を行います．そして LiDAR の点群を IMU 座標に変換し，式 (4.6)，(4.7) に示す残差ベクトルとコスト関数を定めます．そして，残差に対するヤコビアンを求めてコスト関数の最小化を実施しますが，本章で述べる LIO では，式 (5.1) に示す状態を用いて最適化を行うため，求めるヤコビアンは 3.3 節で導出したヤコビアンと異なります．

本章で述べる LIO で求めるヤコビアンは以下となります．

$$\frac{\partial r_i}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial r_i}{\partial {}^O\mathbf{t}_I} & \frac{\partial r_i}{\partial {}^O R_I} & \frac{\partial r_i}{\partial {}^O\mathbf{v}} & \frac{\partial r_i}{\partial \mathbf{b}^\omega} & \frac{\partial r_i}{\partial \mathbf{b}^a} & \frac{\partial r_i}{\partial \mathbf{g}} & \frac{\partial r_i}{\partial {}^I\mathbf{t}_L} & \frac{\partial r_i}{\partial {}^I R_L} \end{pmatrix}^\top \in \mathbb{R}^{1\times 24} \tag{5.6}$$

それぞれのヤコビアンの導出は省きますが，それぞれ以下となります（角速度ベクトルに関するヤコビ

[*1] 何度も計算して確かめてはいますが確実にあっているか自信はありません．

アンの導出は 5.5 節で述べています).

$$
\begin{aligned}
\frac{\partial r_i}{\partial {}^O\mathbf{t}_I} &= -\mathbf{n}_i^\top \\
\frac{\partial r_i}{\partial {}^OR_I} &= \mathbf{n}_i^\top \left({}^OR_I\,{}^I\mathbf{p}_i\right)^\wedge \\
\frac{\partial r_i}{\partial {}^O\mathbf{v}} &= -\Delta t\mathbf{n}_i^\top \\
\frac{\partial r_i}{\partial \mathbf{b}^\omega} &= \mathbf{n}_i^\top \left({}^OR_I\,{}^I\mathbf{p}_i\right)^\wedge J_r\left(\Delta\boldsymbol{\phi}\right)\Delta t \\
\frac{\partial r_i}{\partial \mathbf{b}^a} &= \frac{1}{2}\Delta t^2\mathbf{n}_i^\top {}^OR_{I,t-1} \\
\frac{\partial r_i}{\partial \mathbf{g}} &= -\frac{1}{2}\Delta t^2\mathbf{n}_i^\top \\
\frac{\partial r_i}{\partial {}^I\mathbf{t}_L} &= \mathbf{n}_i^\top {}^OR_I \\
\frac{\partial r_i}{\partial {}^IR_L} &= \mathbf{n}_i^\top {}^OR_I\left({}^IR_L\,{}^L\mathbf{p}_i\right)^\wedge
\end{aligned}
\tag{5.7}
$$

ここで ${}^L\mathbf{p}$ と ${}^I\mathbf{p}$ は同じ点を LiDAR，および IMU 座標で表したものであり ${}^I\mathbf{p} = {}^IT_L\,{}^L\mathbf{p}$ となります．また $\Delta\boldsymbol{\phi} = \left(\boldsymbol{\omega}_t - \mathbf{b}_{t-1}^\omega\right)\Delta t$ であり，これは状態量である $\mathbf{b}_{t-1}^\omega$ が変更される度に再計算します．

残差，およびヤコビアンを求めることができたら，Iterated Extended Kalman Filter（IEKF）を用いた更新を行いますが，これは次式に従い状態の反復更新を行うことで達成されます．

$$
\begin{aligned}
\mathbf{r}^k &= \begin{pmatrix} r_1^k & \cdots & r_N^k \end{pmatrix}^\top \\
J^k &= \begin{pmatrix} J_1^k & \cdots & J_N^k \end{pmatrix}^\top \\
H^k &= \begin{pmatrix}
I_3 & 0_{3\times3} & 0_{3\times15} & 0_{3\times3} \\
0_{3\times3} & J_l^{-1}\left(\left(\log\left(\left({}^OR_I^1\right)^\top {}^OR_I^k\right)\right)^\vee\right) & 0_{3\times15} & 0_{3\times3} \\
0_{15\times3} & 0_{15\times3} & I_{15\times15} & 0_{3\times3} \\
0_{3\times3} & 0_{3\times3} & 0_{3\times3} & J_l^{-1}\left(\left(\log\left(\left({}^IR_L^1\right)^\top {}^IR_L^k\right)\right)^\vee\right)
\end{pmatrix} \\
\bar{\Sigma}^k &= \left(H^k\right)^{-1}\Sigma\left(\left(H^k\right)^{-1}\right)^\top \\
K^k &= \left(\left(J^k\right)^\top R^{-1}J^k + \left(\bar{\Sigma}^k\right)^{-1}\right)^{-1}\left(J^k\right)^\top R^{-1} \\
\mathbf{x}^{k+1} &= \mathbf{x}^k \boxplus \left(-K^k\mathbf{r}^k - \left(I_{24} - K^kJ^k\right)\left(H^k\right)^{-1}\left(\mathbf{x}^k \boxminus \mathbf{x}^1\right)\right)
\end{aligned}
\tag{5.8}
$$

ここで $R = \mathrm{diag}\left(\sigma^2, \cdots \sigma^2\right) \in \mathbb{R}^{N\times N}$ であり，$\sigma^2 \in \mathbb{R}$ は残差に対する分散になります．また，右上に 1 のつく状態は，反復計算を行う際のそれぞれの初期値になります．そして，k 回目の反復計算で更新量が一定以下となり収束したと判定されたら，以下のように状態と共分散行列を更新します．

$$
\begin{aligned}
\mathbf{x}_t &= \mathbf{x}^k \\
\Sigma_t &= \left(I_{24} - K^kJ^k\right)\bar{\Sigma}^k
\end{aligned}
\tag{5.9}
$$

5.4 実用にあたって

前章で述べたルーズカップリングに基づく LIO では，重力加速度や LiDAR-IMU 間の相対姿勢などは推定対象に含まれていませんでした．一方でタイトカップリングに基づく手法では，これらのパラ

メータも同時に最適化することが可能です．これは，ルーズカップリングのように推定を 2 段階に分ける方法では冗長性が生じるのに対し，タイトカップリングではそのような冗長性を排除し，すべてのパラメータを統一的に推定できるためです．

ただし，これらのパラメータを推定に含めたからといって，常に劇的な性能向上が得られるわけではありません．しかし多くの場面において，タイトカップリングを用いた方がより高精度な推定結果が得られる傾向があります．なお，重力加速度や LiDAR-IMU 間の相対姿勢を最適化しなくとも，タイトカップリングに基づく LIO は十分に機能するため，これらのパラメータは必ずしも最適化されるべきとは限りません．

本書では扱いませんが，近年では LiDAR，カメラ，IMU を併用して最適化を行う手法も提案されています．このような手法においては，LiDAR-カメラ間の相対姿勢を高精度に求めることが極めて重要となります．しかし，これらの外部パラメータは事前のキャリブレーションのみでは十分に正確に求められないことも多いため，タイトカップリングの枠組みの中で LiDAR-カメラ間の相対姿勢を同時に最適化することが，実践的かつ有効なアプローチといえるます．

また，IEKF には少し面白い性質があります．式 (5.6) に残差に対するヤコビアンを示していますが，この中には少し複雑なヤコビアンが表れます．例えば速度ベクトルや IMU の計測バイアスに関するヤコビアンは，IMU プレインテグレーションによる動作まで考慮して連鎖則を用いて求める必要があります．実装ではこれらを求めてヤコビアンとして利用していますが，実はこれらのヤコビアンを全て $\mathbf{0}_3$ にしたとしてもタイトカップリングとして機能し，速度やバイアスを求めることが可能です．これは式 (5.8) に示すカルマンゲイン K を介して，ヤコビアンを求めていない状態にも修正量が伝播されるためです．そのため，ヤコビアンの計算が煩雑，また IMU プレインテグレーションに基づく再計算を除外したい場合などは，ヤコビアンを求めなくとも機能させることができます．

またタイトカップリングを用いた LIO の実装として，**因子グラフ**（Factor Graph）内で IMU プレインテグレーションファクタを用いる方法もあります．LIO-SAM [**?**] や GLIM [**?**] ではこの方式が採用されており，この方法を用いると過去の系列も考慮しながら，地図のスムージングなども実装できます．ただし一般的には，逐次処理を行う IEKF による実装のほうが計算コストが低くなることが多いです（LIO-SAM や GLIM も十分な計算速度で実行可能です）．

5.5　回転および角速度バイアスに関するヤコビアンの導出

式 (5.4)，(5.5) に共分散行列を更新するために用いられるヤコビアンを示していますが，回転に関するヤコビアンは導出が複雑なので，本説で補足としてそれらの解説をします．

まず，${}^{O}\mathbf{t}_{I,t}$ の ${}^{O}R_{I,t-1}$ に関するヤコビアンを求めます．これは，微小摂動した ${}^{O}R_{I,t-1}$，すなわち $\exp\left(\delta\boldsymbol{\phi}^{\wedge}\right){}^{O}R_{I,t-1}$ を含む ${}^{O}\mathbf{t}_{I,t}$ の差分を考えることで導けます．表記を簡略化するために，$\exp\left(\delta\boldsymbol{\phi}^{\wedge}\right){}^{O}R_{I,t-1}$ を含む ${}^{O}\mathbf{t}_{I,t}$ を ${}^{O}\mathbf{t}_{I,t}\left(\delta\boldsymbol{\phi}\right)$ とし，式 (5.2) に示す ${}^{O}\mathbf{t}_{I,t}$ との差分を考え，${}^{O}\mathbf{t}_{I,t}\left(\delta\boldsymbol{\phi}\right) - {}^{O}\mathbf{t}_{I,t} = J\delta\boldsymbol{\phi}$ となる J を求めます．

$$
\begin{aligned}
{}^{O}\mathbf{t}_{I,t}\left(\delta\boldsymbol{\phi}\right) - {}^{O}\mathbf{t}_{I,t} =& \frac{1}{2}\left(\exp\left(\delta\boldsymbol{\phi}^{\wedge}\right) - I_3\right){}^{O}R_{I,t-1}\hat{\mathbf{a}}_t\Delta t^2 \\
=& \frac{1}{2}\delta\boldsymbol{\phi}^{\wedge}{}^{O}R_{I,t-1}\hat{\mathbf{a}}_t\Delta t^2 \\
=& -\frac{1}{2}\left({}^{O}R_{I,t-1}\hat{\mathbf{a}}_t\right)^{\wedge}\Delta t^2\delta\boldsymbol{\phi}
\end{aligned}
\tag{5.10}
$$

よって $J = -\frac{1}{2}\left({}^{O}R_{I,t-1}\hat{\mathbf{a}}\right)^{\wedge}\Delta t^2$ となります．${}^{O}\mathbf{v}_t$ の ${}^{O}R_{I,t-1}$ に関するヤコビアンは同様の計算で求

めることができます．

次に ${}^{O}R_{I,t}$ のヤコビアンを考えますが，まず，回転に関する更新を以下に再掲します．

$$R_t = R_{t-1} \exp\left(\left(\boldsymbol{\omega}_t - \mathbf{b}^{\omega}_{t-1} - \boldsymbol{\eta}^{\omega}_t\right)^{\wedge} \Delta t\right) \tag{5.11}$$

R_t に関する共分散行列は，これに対応する回転ベクトル $\left(\log\left(R_t\right)\right)^{\vee}$ に対して定義されるものになります．そのため，$\left(\log\left(R_t\right)\right)^{\vee}$ に対する R_{t-1} と $\mathbf{b}^{\omega}_{t-1}$ の微分を考える必要があります．

まず R_{t-1} に関するヤコビアンを考えますが，これは以下の連鎖則を用いて計算できます．

$$\frac{\partial \left(\log\left(R_t\right)\right)^{\vee}}{\partial R_{t-1}} = \frac{\partial \left(\log\left(R_t\right)\right)^{\vee}}{\partial R_t} \frac{\partial R_t}{\partial R_{t-1}} \tag{5.12}$$

まず $\partial \left(\log\left(R_t\right)\right)^{\vee} / \partial R_t$ を考えるにあたり，次式を満たすヤコビアンを考えます．

$$\left(\log\left(\exp\left(\delta\boldsymbol{\phi}^{\wedge}\right) R_t\right)\right)^{\vee} - \left(\log\left(R_t\right)\right)^{\vee} = J\delta\boldsymbol{\phi} \tag{5.13}$$

左辺第一項の BCH 展開を考えると，1 次近似として $\left(\log\left(\exp\left(\delta\boldsymbol{\phi}^{\wedge}\right) R_t\right)\right)^{\vee} \simeq \left(\log\left(R_t\right)\right)^{\vee} + J_l^{-1}\left(\left(\log\left(R_t\right)\right)^{\vee}\right)\delta\boldsymbol{\phi}$ が得られるため，上式を満たすヤコビアンは以下となります．

$$J_l^{-1}\left(\left(\log\left(R_t\right)\right)^{\vee}\right) \tag{5.14}$$

次に $\partial R_t / \partial R_{t-1}$ を考えるために，次式を満たす J を考えます．

$$\left(R_{t-1} \exp\left(\left(\hat{\boldsymbol{\omega}}_t\right)^{\wedge} \Delta t\right)\right)^{-1} \exp\left(\delta\boldsymbol{\phi}^{\wedge}\right) R_{t-1} \exp\left(\left(\hat{\boldsymbol{\omega}}_t\right)^{\wedge} \Delta t\right) = I_3 + \left(J\delta\boldsymbol{\phi}\right)^{\wedge} \tag{5.15}$$

ただし，$\hat{\boldsymbol{\omega}}_t = \boldsymbol{\omega}_t - \mathbf{b}^{\omega}_{t-1} - \boldsymbol{\eta}^{\omega}_t$ としています．式 (5.15) の左辺を展開すると以下が得られます．

$$\begin{aligned} &\exp\left(-\left(\hat{\boldsymbol{\omega}}_t\right)^{\wedge} \Delta t\right) R_{t-1}^{-1} \exp\left(\delta\boldsymbol{\phi}^{\wedge}\right) R_{t-1} \exp\left(\left(\hat{\boldsymbol{\omega}}_t\right)^{\wedge} \Delta t\right) \\ =& \exp\left(\left(\mathrm{Ad}_{\exp\left(-\left(\hat{\boldsymbol{\omega}}_t\right)^{\wedge} \Delta t\right) R_{t-1}^{-1}} \delta\boldsymbol{\phi}\right)^{\wedge}\right) \\ \simeq& I_3 + \left(\mathrm{Ad}_{\exp\left(-\left(\hat{\boldsymbol{\omega}}_t\right)^{\wedge} \Delta t\right) R_{t-1}^{-1}} \delta\boldsymbol{\phi}\right)^{\wedge} \end{aligned} \tag{5.16}$$

よって $J = \mathrm{Ad}_{\exp\left(-\left(\hat{\boldsymbol{\omega}}_t\right)^{\wedge} \Delta t\right) R_{t-1}^{-1}}$ であり，これは式 (2.51) を用いると $\exp\left(-\left(\hat{\boldsymbol{\omega}}_t\right)^{\wedge} \Delta t\right) R_{t-1}^{-1}$ となります．また $\exp\left(-\left(\hat{\boldsymbol{\omega}}_t\right)^{\wedge} \Delta t\right) R_{t-1}^{-1}$ は R_t^{-1} と等しいため，$R_t^{\top}$ となります．

以上より，式 (5.12) は以下となります．

$$\frac{\partial \left(\log\left(R_t\right)\right)^{\vee}}{\partial R_{t-1}} = J_l^{-1}\left(\left(\log\left(R_t\right)\right)^{\vee}\right) R_t^{\top} \tag{5.17}$$

次に，$\mathbf{b}^{\omega}_{t-1}$ に関するヤコビアンを考えますが，こちらも連鎖則を用いて以下のように計算できます．

$$\frac{\partial \left(\log\left(R_t\right)\right)^{\vee}}{\partial \mathbf{b}^{\omega}_{t-1}} = \frac{\partial \left(\log\left(R_t\right)\right)^{\vee}}{\partial R_t} \frac{\partial R_t}{\partial \mathbf{b}^{\omega}_{t-1}} \tag{5.18}$$

$\partial \left(\log\left(R_t\right)\right)^{\vee} / \partial R_t$ は式 (5.14) に示されていますので，$\partial R_t / \partial \mathbf{b}^{\omega}_{t-1}$ について考えます．

$\partial R_t / \partial \mathbf{b}^{\omega}_{t-1}$ を求めるために，以下の式を満たす J を考えます．

$$\left(R_{t-1} \exp\left(\left(\hat{\boldsymbol{\omega}}_t\right)^{\wedge} \Delta t\right)\right)^{-1} R_{t-1} \exp\left(\left(\hat{\boldsymbol{\omega}}_t - \delta\mathbf{b}^{\omega}\right)^{\wedge} \Delta t\right) = I_3 + \left(J\delta\mathbf{b}^{\omega}\right)^{\wedge} \tag{5.19}$$

なお式 (5.19) の左辺は $\exp\left(\left(-\hat{\boldsymbol{\omega}}_t\right)^\wedge \Delta t\right) \exp\left(\left(\hat{\boldsymbol{\omega}}_t - \delta\mathbf{b}^\omega\right)^\wedge \Delta t\right)$ となります．ここで BCH 展開を用いると，式 (5.19) の左辺は以下のように一次近似できます．

$$\begin{aligned}\exp\left(\left(-\hat{\boldsymbol{\omega}}_t\right)^\wedge \Delta t\right) \exp\left(\left(\hat{\boldsymbol{\omega}}_t - \delta\mathbf{b}^\omega\right)^\wedge \Delta t\right) \simeq & \exp\left(-\left(J_r\left(\hat{\boldsymbol{\omega}}\Delta t\right)\delta\mathbf{b}^\omega\right)^\wedge \Delta t\right) \\ \simeq & I_3 - \left(J_r\left(\hat{\boldsymbol{\omega}}\Delta t\right)\delta\mathbf{b}^\omega\right)^\wedge \Delta t\end{aligned} \tag{5.20}$$

ここで $J_r(\cdot)$ は SO(3) に関する右ヤコビアンであり以下となります．

$$J_r(\boldsymbol{\phi}) = I_3 - \frac{1-\cos\theta}{\theta^2}\boldsymbol{\phi}^\wedge + \frac{\theta - \sin\theta}{\theta^3}\left(\boldsymbol{\phi}^\wedge\right)^2 \tag{5.21}$$

ただし $\|\boldsymbol{\phi}\|$ です．よって式 (5.19) を満たす J は $-J_r\left(\hat{\boldsymbol{\omega}}\Delta t\right)\Delta t$ となります．

以上より，式 (5.18) は以下となります．

$$\frac{\partial\left(\log\left(R_t\right)\right)^\vee}{\partial\mathbf{b}_{t-1}^\omega} = -J_l^{-1}\left(\left(\log\left(R_t\right)\right)^\vee\right) J_r\left(\hat{\boldsymbol{\omega}}_t\Delta t\right)\Delta t \tag{5.22}$$

また式 (5.6) に示される残差 r_i に対する角速度バイアス $\mathbf{b}^\omega$ に関するヤコビアンは，以下のように連鎖則を用いて計算されます．

$$\frac{\partial r_i}{\partial\mathbf{b}^\omega} = \frac{\partial r_i}{\partial\mathbf{r}_i}\frac{\partial\mathbf{r}_i}{\partial^O\mathbf{t}_I}\frac{\partial^O\mathbf{t}_I}{\partial^O R_I}\frac{\partial^O R_I}{\partial\mathbf{b}^\omega} + \frac{\partial r_i}{\partial\mathbf{r}_i}\frac{\partial\mathbf{r}_i}{\partial^O R_I}\frac{\partial^O R_I}{\partial\mathbf{b}^\omega} \tag{5.23}$$

ここで，${}^O R_I$ と $\mathbf{b}^\omega$ に関するヤコビアンはそれぞれ以下となります．

$$\begin{aligned}\frac{\partial^O\mathbf{t}_I}{\partial^O R_I} &= -\frac{1}{2}\left({}^O R_I\hat{\mathbf{a}}_t\right)^\wedge \Delta t^2 \\ \frac{\partial^O R_I}{\partial\mathbf{b}^\omega} &= J_r\left(\Delta\boldsymbol{\phi}\right)\Delta t\end{aligned} \tag{5.24}$$

これらを踏まえると，式 (5.23) は以下となります．

$$\begin{aligned}\frac{\partial r_i}{\partial\mathbf{b}^\omega} =& \mathbf{n}_i^\top\left(-I_3\right)\left(-\frac{1}{2}\left({}^O R_I\hat{\mathbf{a}}_t\right)^\wedge \Delta t^2\right)\left(J_r\left(\Delta\boldsymbol{\phi}\right)\Delta t\right) + \mathbf{n}_i^\top\left({}^O R_I{}^I\mathbf{p}_i\right)^\wedge\left(J_r\left(\Delta\boldsymbol{\phi}\right)\Delta t\right) \\ \simeq& \mathbf{n}_i^\top\left({}^O R_I{}^I\mathbf{p}_i\right)^\wedge J_r\left(\Delta\boldsymbol{\phi}\right)\Delta t\end{aligned} \tag{5.25}$$

ただし右辺第一項は，Δt^3 が含まれるため微小量として無視しました．

第 6 章

グラフベース SLAM

前章までは LIO について説明しました．LIO を用いることで移動量を求めることができますが，移動量推定には基本的にはドリフト誤差が含まれるため，LIO で求められた移動量を基に LiDAR の点群データを貼り合わせただけでは，正確な地図を構築することはできません．正確な地図を得るためには，このドリフト誤差を補正し，点群を貼り合わせる必要があります．本書では，これを実現するために**グラフベース SLAM**（Graph-based SLAM）を用います．

6.1　グラフベース SLAM の流れ

グラフベース SLAM の実装には様々なものがありますが，本書で想定する実装の流れを最初に示します．まず，オドメトリとマッピングの 2 つのプロセスを並列で起動させます．マッピングプロセスは，オドメトリにより推定されたオドメトリ座標上の IMU の姿勢 ${}^{O}T_I \in \mathrm{SE}(3)$，およびこれに対応する LiDAR の計測点群 ${}^{I}\mathcal{P}$ を受け取ります．オドメトリとしては，前章までで述べられたものが利用さることになりますので，本章ではマッピングのプロセスが行う処理の解説をします．

グラフベース SLAM におけるマッピングプロセスでは，ノード集合 $\mathcal{V}$ とエッジ集合 $\mathcal{E}$ で構成されるグラフの作成を行います．ここでノードとは，センサの姿勢やランドマークの位置を表すものであり，エッジはこれらのノード間の相対姿勢を表します．ただし本書で示す実装では，ノードを用いて表すのはセンサの姿勢のみとします．このようなグラフは**ポーズグラフ**（Pose Graph）と呼ばれます．本書で紹介するマッピングプロセスでは，このポーズグラフの構築を行い，ポーズグラフに対して定まるコスト関数の最適化を行います．

ポーズグラフ構築のために，マッピングプロセスでは，オドメトリプロセスから姿勢 ${}^{O}T_I$ を受け取り，これを用いて移動量を計算していきます．そして移動量がある一定値を超えた場合にその姿勢をキーフレーム ${}^{M}T_I$ として検出します[*1]．${}^{M}T_{I,i}$ が最新のキーフレームとして検出されると，キーフレームとそれに対応する LiDAR の計測点群 ${}^{I}\mathcal{P}_i$ を保存します．また，オドメトリにより計算された姿勢に基づいて，2 つのキーフレーム間のエッジ（オドメトリエッジ）を以下のように定めます．

$$E_{i-1,i} = {}^{O}T_{I,i-1}^{-1}\,{}^{O}T_{I,i} \tag{6.1}$$

オドメトリエッジを計算した後に，**ループ検知**（Loop Detection）を行います．ループ検知とは，現在いる地点が過去に通過したことがある地点であるかどうかを識別し，過去に通過した地点であると識

[*1] キーフレームの検出にも様々な工夫をすることができますが，本書ではシンプルに移動量に対して閾値を設け，一定量移動する毎にキーフレームとして検出していきます．

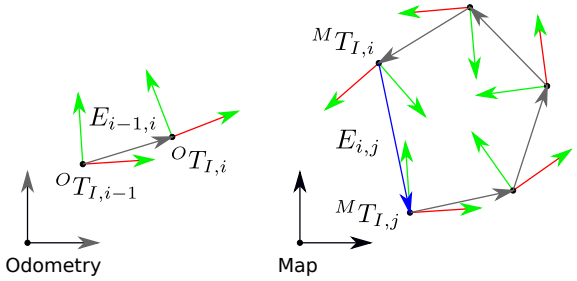

図 6.1　Residuals used in pose graph optimization.

別されれば，今の姿勢と過去に通過した際の姿勢の間相対姿勢を認識することです．ループ検知も様々な方法を用いて実行することが可能ですが，本書で用いる方法ではまず，新たにキーフレームとして検知された姿勢から最も近い N 個のキーフレームを選択します．キーフレーム間の近さを測る際には，並進ベクトルのみを用います．そして，新しく追加されたキーフレーム，および過去に検出されたキーフレームに対応するそれぞれの LiDAR の計測点群を用いて，スキャンマッチングを行います．ループ検知におけるスキャンマッチングでは，速度より精度が重視されるため，少し計算コストが大きくなりますが，Generalized ICP（GICP）[**?**] を用います．GICP の詳細は 6.3 節で述べます．GICP によるスキャンマッチングが成功したと判定された場合に，ループ検知が行われたとしてループエッジの追加を行います．

今，最新のキーフレーム ${}^{M}T'_{I,i}$ に対して，j 番目のキーフレームとの GICP の結果が以下のようになったとします．

$$ {}^{M}T_{I,i} = \underset{{}^{M}T'_{I,i}}{\operatorname{argmin}} E_{\mathrm{GICP}}({}^{M}T'_{I,i}; {}^{M}T_{I,j}, {}^{I}\mathcal{P}_i, {}^{I}\mathcal{P}_j) \tag{6.2} $$

ここで $E_{\mathrm{GICP}}(\cdot)$ は GICP のコスト関数であり，${}^{M}T_{I,j}$，${}^{I}\mathcal{P}_i$，${}^{I}\mathcal{P}_j$ は不変であるとしています．式 (6.2) により得られた姿勢 ${}^{M}T_{I,i}$ を用いて，ループエッジを以下のように定めます．

$$ E_{i,j} = {}^{M}T_{I,i}^{-1}{}^{M}T_{I,j} \tag{6.3} $$

ループ検知を行いループエッジがエッジ集合に新たに追加されると，ポーズフラフの最適化を行います．ポーズグラフの最適化に関しては次節で述べます．なお，オドメトリエッジとループエッジの簡単な概略を図 6.1 に図示しています．

6.2　ポーズグラフの最適化

6.2.1　コスト関数

ポーズグラフの最適化を考えるために，ポーズグラフ内で定義される残差ベクトルについて考えます．式 (6.1)，(6.3) に示すように，ポーズグラフにおけるエッジは 2 つの姿勢間の相対姿勢として定義されます．このエッジが観測，すなわち固定値であると仮定し，以下の残差ベクトルを定義します．

$$ \mathbf{r}_{i,j}({}^{M}T_{I,i}, {}^{M}T_{I,j}) = \left(\log\left(E_{i,j}^{-1}{}^{M}T_{I,i}^{-1}{}^{M}T_{I,j}\right)\right)^{\vee} \tag{6.4} $$

式 (6.4) が i 番目と j 番目のエッジに対して定まる残差ベクトルとなるため，この総和を最小化する姿勢の集合を求めることでポーズグラフの最適化を行います．

$$E\left(\mathcal{T}\right) = \sum_{i,j\in\mathcal{E}} \rho\left(\left\|\mathbf{r}_{i,j}\right\|^2_{\Omega_{i,j}}\right) \tag{6.5}$$

なお $\left\|\mathbf{r}_{i,j}\right\|^2_{\Omega_{i,j}} = \mathbf{r}_{i,j}^\top \Omega_{i,j} \mathbf{r}_{i,j}$ は情報行列 $\Omega_{i,j}$ を用いたマハラノビス距離の 2 乗を表します．

6.2.2 ヤコビアンの計算

式 (6.5) のコスト関数を最小化するにあたり，式 (6.4) に示す残差ベクトルの ${}^M T_{I,i}$ と ${}^M T_{I,j}$ に関するヤコビアンを求めます．ヤコビアンの計算のために，$\Delta_{i,j} = E_{i,j}^{-1} {}^M T_{I,i}^{-1} {}^M T_{I,j}$ を導入します．$\Delta_{i,j}$ を用いると，ヤコビアンはそれぞれ連鎖則を用いて以下のように計算できます．

$$\begin{aligned}
\frac{\partial \mathbf{r}_{i,j}}{\partial {}^M T_{I,i}} &= \frac{\partial \mathbf{r}_{i,j}}{\partial \Delta_{i,j}} \frac{\partial \Delta_{i,j}}{\partial {}^M T_{I,i}} \\
\frac{\partial \mathbf{r}_{i,j}}{\partial {}^M T_{I,j}} &= \frac{\partial \mathbf{r}_{i,j}}{\partial \Delta_{i,j}} \frac{\partial \Delta_{i,j}}{\partial {}^M T_{I,j}}
\end{aligned} \tag{6.6}$$

以下，それぞれの微分について考えます．

まず $\partial \mathbf{r}_{i,j} / \partial \Delta_{i,j}$ を考えます．式 (6.4) より，以下の等式が成り立つ J を求めれば良いことになります．

$$\left(\log\left(\exp\left(\delta\boldsymbol{\xi}^\wedge\right)\Delta_{i,j}\right)\right)^\vee - \left(\log\left(\Delta_{i,j}\right)\right)^\vee = J\delta\boldsymbol{\xi} \tag{6.7}$$

ここで $\log\left(\exp\left(\delta\boldsymbol{\xi}^\wedge\right)\Delta_{i,j}\right)$ は，BCH 展開を用いて以下のように 1 次近似することができます．

$$\left(\log\left(\exp\left(\delta\boldsymbol{\xi}^\wedge\right)\Delta_{i,j}\right)\right)^\vee \simeq \mathbf{r}_{i,j} + J_l^{-1}\left(\mathbf{r}_{i,j}\right)\delta\boldsymbol{\xi} \tag{6.8}$$

なお，$\mathbf{r}_{i,j} = \left(\log\left(\Delta_{i,j}\right)\right)^\vee$ であり，$J_l(\cdot)$ は SE(3) に関する左ヤコビアンと呼ばれ，以下のように定義されます．

$$\begin{aligned}
J_l(\boldsymbol{\xi}) &= \begin{pmatrix} J_l(\boldsymbol{\phi}) & 0_{3\times 3} \\ Q(\boldsymbol{\xi}) & J_l(\boldsymbol{\phi}) \end{pmatrix} \\
J_l(\boldsymbol{\phi}) &= I_3 + \frac{1-\cos\theta}{\|\theta\|^2}\boldsymbol{\phi}^\wedge + \frac{\theta-\sin\theta}{\|\theta\|^3}\left(\boldsymbol{\phi}^\wedge\right)^2 \\
Q(\boldsymbol{\xi}) &= \frac{1}{2}\mathbf{v}^\wedge + \frac{1-\alpha}{\|\boldsymbol{\phi}\|^2}\left(\boldsymbol{\phi}^\wedge\mathbf{v}^\wedge + \mathbf{v}^\wedge\boldsymbol{\phi}^\wedge\right) + \frac{\beta-1}{\|\boldsymbol{\phi}\|^4}\boldsymbol{\phi}^\wedge\mathbf{v}^\wedge\boldsymbol{\phi}^\wedge \\
\alpha &= \frac{\sin\theta}{\theta}\cdot\frac{\theta+\sin\theta}{2\left(1-\cos\theta\right)} \\
\beta &= \frac{1}{\theta^2}\left(1-\frac{\sin\theta}{\theta}\right)
\end{aligned} \tag{6.9}$$

ただし，$\boldsymbol{\xi}^\wedge = \begin{pmatrix} \begin{pmatrix} \mathbf{v}^\top & \boldsymbol{\phi}^\top \end{pmatrix}^\top \end{pmatrix}^\wedge \in \mathfrak{se}(3)$, $\boldsymbol{\phi}^\wedge \in \mathfrak{so}(3)$, $\theta = \|\boldsymbol{\phi}\|$ です[*2]．式 (6.7), (6.8) から，$\partial \mathbf{r}_{i,j} / \partial \Delta_{i,j}$ は以下になります．

$$\frac{\partial \mathbf{r}_{i,j}}{\partial \Delta_{i,j}} = J_l^{-1}(\mathbf{r}_{i,j}) \tag{6.10}$$

[*2] 式 (6.9) には SE(3) と SO(3) の左ヤコビアンがどちらも記載されていますが，引数が $\mathfrak{se}(3)$ か $\mathfrak{so}(3)$ でそれぞれを区別します．

次に $\partial\Delta_{i,j}/\partial {}^{M}T_{I,i}$ の微分を考えるために，まず $\Delta_{i,j}\left(\delta\boldsymbol{\xi}\right) = E_{i,j}^{-1}\left(\exp\left(\delta\boldsymbol{\xi}^{\wedge}\right){}^{M}T_{I,i}\right)^{-1}{}^{M}T_{I,j}$, $\delta\boldsymbol{\xi}^{\wedge} \in \mathfrak{se}(3)$ を導入します．この $\Delta_{i,j}\left(\delta\boldsymbol{\xi}\right)$ に左から $\Delta_{i,j}^{-1}$ を掛けることで，恒等元 I_4 の周辺での変化量を考えます．

$$\begin{aligned}\Delta_{i,j}^{-1}\Delta_{i,j}\left(\delta\boldsymbol{\xi}\right) =& {}^{M}T_{I,j}^{-1}{}^{M}T_{I,i}E_{i,j}E_{i,j}^{-1}{}^{M}T_{I,i}^{-1}\exp\left(-\delta\boldsymbol{\xi}^{\wedge}\right){}^{M}T_{I,j}\\ =& {}^{M}T_{I,j}^{-1}\exp\left(-\delta\boldsymbol{\xi}\right){}^{M}T_{I,j}\\ =& \exp\left(\left(-\operatorname{Ad}_{{}^{M}T_{I,j}^{-1}}\delta\boldsymbol{\xi}\right)^{\wedge}\right)\\ \simeq& I_4 + \left(-\operatorname{Ad}_{{}^{M}T_{I,j}^{-1}}\delta\boldsymbol{\xi}\right)^{\wedge}\end{aligned} \tag{6.11}$$

式 (2.48) と比較すると，$\partial\Delta_{ij}/\partial {}^{M}T_{I,i}$ は以下になることがわかります．

$$\frac{\partial\Delta_{i,j}}{\partial {}^{M}T_{I,i}} = -\operatorname{Ad}_{{}^{M}T_{I,j}^{-1}} \tag{6.12}$$

以上から，式 (6.10)，(6.12) より，$\partial\mathbf{r}_{i,j}/\partial {}^{M}T_{I,i}$ は以下となります．

$$\frac{\partial\mathbf{r}_{i,j}}{\partial {}^{M}T_{I,i}} = -J_l^{-1}(\mathbf{r}_{i,j})\operatorname{Ad}_{{}^{M}T_{I,j}^{-1}} \tag{6.13}$$

なお同様の計算を行うと，$\partial\mathbf{r}_{i,j}/\partial {}^{M}T_{I,j}$ を以下のように導くことができます．

$$\frac{\partial\mathbf{r}_{i,j}}{\partial {}^{M}T_{I,j}} = J_l^{-1}(\mathbf{r}_{i,j})\operatorname{Ad}_{{}^{M}T_{I,i}} \tag{6.14}$$

6.2.3　ガウス・ニュートン法による最適化

式 (6.5) に示すコスト関数を最小化するためにも，ガウス・ニュートン法を用います．まず式 (6.4) に示す残差ベクトル $\mathbf{r}_{i,j}$ に対する ${}^{M}T_{I,i}$，および ${}^{M}T_{I,j}$ に関するヤコビアンは，それぞれ式 (6.13)，(6.14) のように定まります．以下，それぞれのヤコビアンを J_i，J_j とします．今，i,j 番目のエッジに対して定まる残差ベクトル $\mathbf{r}_{i,j}$ に対するヤコビアンを $J_{i,j} = (0\ \cdots\ 0\ J_i\ 0\ \cdots 0\ J_j\ 0\ \cdots\ 0) \in \mathbb{R}^{6\times 6N}$ とします（i,j 番目の残差に影響を与えるのは i，j 番目の姿勢のみのため，それ以外の要素はすべて 0 になります．）．これを用いて，ヘッセ行列 $H \in \mathbb{R}^{6N\times 6N}$，と勾配ベクトル $\mathbf{b} \in \mathbb{R}^{6N}$ を求めます．

$$\begin{aligned}H &= \sum_{ij\in\mathcal{E}} w_{i,j}J_{i,j}^{\top}J_{i,j}\\ \mathbf{b} &= \sum_{ij\in\mathcal{E}} w_{i,j}J_{i,j}^{\top}\mathbf{r}_{i,j}\end{aligned} \tag{6.15}$$

なお，$w_{i,j} = \rho'\left(\|\mathbf{r}_{i,j}\|_{\Omega_{i,j}^2}\right)$ です．このヘッセ行列と勾配を用いて，$H\delta\boldsymbol{\xi} = -\mathbf{b}$ を満たす $\delta\boldsymbol{\xi}$ を求めます．$\delta\boldsymbol{\xi} \in \mathbb{R}^{6N}$ は N 個のブロックで構成されるベクトルとなっており，各ブロックはそれぞれの姿勢に対応する更新量となっています．つまり i 番目のブロックのベクトルを $\delta\boldsymbol{\xi}_i \in \mathbb{R}^6$ とすると，対応する i 番目の姿勢 ${}^{M}T_{I,i}$ は以下のように更新されます．

$${}^{M}T_{I,i} \leftarrow \exp\left(\delta\boldsymbol{\xi}_i^{\wedge}\right){}^{M}T_{I,i} \tag{6.16}$$

グラフベース SLAM を実装するにあたり，ヘッセ行列のサイズは $6N \times 6N$ となり，単純に逆行列を求めると計算コストが非常に大きくなってしまいます．しかし多くの場合，一般的なグラフベース SLAM では，H の対角成分以外のほとんどの成分が 0 となります．このような行列は**疎行列**（Sparse

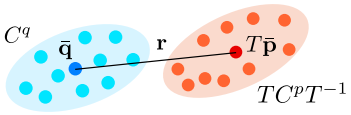

図 6.2　Residual error used in GICP.

Matrix）と呼ばれ，専用のソルバーを用いると高速に $H\delta\boldsymbol{\xi} = -\mathbf{b}$ を満たす $\delta\mathbf{x}$ を求めることができます．そのため，実際には式 (6.15) に示すような大きなサイズのヤコビアンの計算は行わず，J_i，J_j を計算して，行列の各要素に加算していくような計算を行います．

6.3　ループ検知

6.3.1　GICP の最適化

前述の通り，ループ検知のためには GICP を用います．GICP は Source 点群 $\mathcal{P}$ と Target 点群 $\mathcal{Q}$ の 2 つの点群を照合するために利用できる手法です．GICP では，それぞれの点群を正規分布を用いて表現しますが，そのためにまず，各点群の全ての点に対して最近某探索を行い，各点群内から周辺の点を取得します．そしてそられの点を用いて，平均と共分散行列を計算します．平均と分散の計算は式 (3.5)，(3.6) に基づきます．すなわち，$\mathcal{P}$ の i 番目の点は $\bar{\mathbf{p}}_i$ と C_i^p，$\mathcal{Q}$ の i 番目の点は $\bar{\mathbf{q}}_i$ と C_i^q によりそれぞれ表されます．GICP では，これらを用いてコスト関数を以下のように定めます．

$$E\left(T\right) = \sum_{i=1}^{N} \rho\left(\left(\bar{\mathbf{q}}_i - T\bar{\mathbf{p}}_i\right)^\top \left(\hat{C}_i^q + T\hat{C}_i^p T^{-1}\right)^{-1} \left(\bar{\mathbf{q}}_i - T\bar{\mathbf{p}}_i\right)\right) \tag{6.17}$$

ただし $\hat{C}^p$，$\hat{C}^q$ は以下となります．

$$\begin{aligned} \hat{C}^p &= \begin{pmatrix} C^p & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{pmatrix} \in \mathbb{R}^{4\times 4} \\ \hat{C}^q &= \begin{pmatrix} C^q & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{pmatrix} \in \mathbb{R}^{4\times 4} \end{aligned} \tag{6.18}$$

本書では，GICP もガウス・ニュートン法を用いて解きますが，表記の簡略化のために，$\mathbf{r}_i = \bar{\mathbf{q}}_i - T\bar{\mathbf{p}}_i$，$\Omega_i = \left(\hat{C}_i^q + T\hat{C}_i^p T^\top\right)^{-1}$ とおきます．まず $\mathbf{r}_i$ の T に対するヤコビアンを求めるために，以下を満たす J_i を求めます．

$$\bar{\mathbf{q}}_i - \exp\left(\delta\boldsymbol{\xi}^\wedge\right) T\bar{\mathbf{p}}_i - \left(\bar{\mathbf{q}}_i - T\bar{\mathbf{p}}_i\right) = J_i\delta\boldsymbol{\xi} \tag{6.19}$$

式 (6.19) の左辺を展開すると，以下が得られます．

$$\begin{aligned} -\exp\left(\delta\boldsymbol{\xi}^\wedge\right) T\bar{\mathbf{p}}_i + T\bar{\mathbf{p}}_i &= -\left(\exp\left(\delta\boldsymbol{\xi}^\wedge\right) - I_4\right) T\bar{\mathbf{p}}_i \\ &= -\begin{pmatrix} \delta\boldsymbol{\phi}^\wedge & \delta\mathbf{v} \\ \mathbf{0}^\top & 0 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{p}}_i^{'} \\ 1 \end{pmatrix} \\ &= -\begin{pmatrix} -\left(\bar{\mathbf{p}}_i^{'}\right)^\wedge \delta\boldsymbol{\phi} + \delta\mathbf{v} \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} -I_3 & \left(\bar{\mathbf{p}}_i^{'}\right)^\wedge \\ \mathbf{0}^\top & \mathbf{0}^\top \end{pmatrix} \delta\boldsymbol{\xi} \end{aligned} \tag{6.20}$$

ただし，$\bar{\mathbf{p}}_i^{'} = R\bar{\mathbf{p}}_i + \mathbf{t}$ としています．よって，式 (6.19) を満たす J_i は以下になります．

$$J_i = \begin{pmatrix} -I_3 & \left(\bar{\mathbf{p}}_i^{'}\right)^{\wedge} \\ \mathbf{0}^{\top} & \mathbf{0}^{\top} \end{pmatrix} \tag{6.21}$$

このヤコビアンを用いて，ヘッセ行列と勾配を以下のように求めます．

$$\begin{aligned} H &= \sum_{i=1} w_i J_i^{\top} \Omega_i J_i \\ \mathbf{b} &= \sum_{i=1} w_i J_i^{\top} \Omega_i \mathbf{r}_i \end{aligned} \tag{6.22}$$

なお $w_i = \rho'\left(\mathbf{r}_i \Omega_i \mathbf{r}_i\right)$ です．この H と $\mathbf{b}$ を用いて，$H\delta\boldsymbol{\xi} = -\mathbf{b}$ を満たす $\delta\boldsymbol{\xi}$ を求め，$T \leftarrow \exp\left(\delta\boldsymbol{\xi}^{\wedge}\right) T$ で姿勢を更新します．

式 (6.22) の導出について簡単に述べますが，これは 2.3 節でも述べたように，状態が微小変化した際の残差ベクトルの線形近似を用いたコスト関数について考えることで導出できます．

$$\sum_{i=1}^{N} \left(\mathbf{r}_i + J_i\delta\boldsymbol{\xi}\right)^{\top} \Omega_i \left(\mathbf{r}_i + J_i\delta\boldsymbol{\xi}\right) = \sum_{i=1}^{N} \mathbf{r}_i^{\top}\Omega_i\mathbf{r}_i + 2\delta\boldsymbol{\xi}^{\top}\Omega_i J_i\mathbf{r}_i + \delta\boldsymbol{\xi}^{\top} J_i^{\top}\Omega_i J_i\delta\boldsymbol{\xi} \tag{6.23}$$

この右辺を $\delta\boldsymbol{\xi}$ で微分して $\mathbf{0}$ とすると以下が得られます（ただし式 (6.22) と比較して，下の式では w_i が抜けていることに注意してください）．

$$\sum_{i=1}^{N} J_i^{\top}\Omega_i J_i\delta\boldsymbol{\xi} = -\sum_{i=1}^{N} J_i^{\top}\Omega_i\mathbf{r}_i \tag{6.24}$$

6.3.2　ループ検知の成功判定

ループ検知のために GICP を用いて，式 (6.17) に示すコスト関数の最小化を行い，点群の照合を行います．この照合結果を基にループ検知の成功・失敗を判断する必要があるのですが，点群の照合の是非を明示的に示す指標を作成することは困難です．例えば，最終的なコスト関数の値，もしくは残差の平均値が一定以下になっているというような閾値は，ある程度の精度で照合の是非を判定できますが，必ずしも判断の正しい指標になるとはいえません．しかし実装においては，これらの指標を用いるのが有用な場合が多くあります．本書で紹介している実装では，残差の平均値や，照合率（最近傍点との距離が閾値以下の点の割合）などに閾値を定め，これらに基づきループ検知が成功したかどうかの判定を行っています．

ただしこのような方法では，照合に失敗した場合でも成功したと判定してしまう場合があります．このような誤った情報がグラフに組み込まれると，グラフの最適化が破綻してしまうこともあります．そのため式 (6.15) に示すように，グラフの最適化においてもフーバー損失等のロバストカーネルを利用することが有効になります．

6.4　座標変換の計算と地図構築

6.4.1　座標変換

LIO を用いると，オドメトリ座標上での IMU の姿勢 ${}^{O}T_I$ が得られます．またポーズグラフの最適化を実行すると，新しく検出されたキーフレームの位置が修正され，それが地図座標上での IMU の姿勢

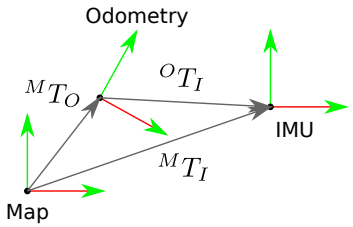

図 6.3　Residuals used in pose graph optimization.

${}^{M}T_{I}$ となります．この 2 つの姿勢を用いて，図 2.1 に示す地図とオドメトリ座標間の変換 ${}^{M}T_{O}$ を求めるにあたり，図 6.3 ように，地図座標上の IMU の姿勢，およびオドメトリ座標上での IMU の姿勢が一致するとします．するとこの関係から，${}^{M}T_{O}$ を以下のように求められます．

$$\begin{aligned} {}^{M}T_{O} &= {}^{M}T_{I}\,{}^{O}T_{I}^{-1} \\ &= {}^{M}T_{I}\,{}^{I}T_{O} \end{aligned} \tag{6.25}$$

なお本書で詳細は述べませんが，自己位置推定をする場合でも同じように ${}^{M}T_{O}$ を求めます．

6.4.2　地図構築

ポーズグラフの最適化が終わると，地図座標上での各キーフレームが ${}^{M}T_{I,i}$ 修正されます．これらの姿勢，および各キーフレームに対応する点群 ${}^{I}\mathcal{P}_{i}$ を用いて，以下のように点群地図を構築します．

$${}^{M}\mathcal{M} = \bigcup_{i=1}^{K} \bigcup_{{}^{I}\mathbf{p} \in {}^{I}\mathcal{P}_{i}} {}^{M}T_{I,i}\,{}^{I}\mathbf{p} \tag{6.26}$$

ここで K はキーフレームの数です．ただし本書で紹介しているグラフベース SLAM では，この地図点群が処理に利用されることはありません．そのため，SLAM プロセス終了時に一度実行するだけでも問題はありません．ポーズグラフの最適化後に都度実行すると，地図が正しく構築できているかをオンラインで確認することができるため，本書で紹介する実装では最適化後に毎回実行しています．

6.5　実用にあたって

本章で紹介した方法では，移動量に対して閾値を設けてキーフレームを検出し，そのキーフレームに対応する LiDAR の観測点群を保存しておき，この点群をキーフレームに合わせて座標変換することで地図の作成を行いました．この方法を採用すると，LiDAR が計測した全ての点群を用いて地図の作成を行わくなるため，地図点群の密度が低下するという問題が発生してしまいます．点群の密度を上げるために，例えば一定時間 LiDAR の計測点群を蓄積したらキーフレームとして検出するという方法も考えられますが，この方法を用いると時間経過とともに使用するメモリ容量が増大してしまいます．一方で移動量に対して閾値を設けてキーフレームを検出する方法であれば，移動量に対してメモリ容量が増大するため，メモリ効率は優れるというトレードオフがあります．

もし単に点群密度の高い地図を作りたいというだけであれば，SLAM を用いて地図を作成した際に使用した同じデータを用いて，その地図上で再度位置推定を実行し，その位置推定結果を基に点群をマッピングするという方法もあります．少し面倒にはなりますが，この方法であればメモリコストを気にせずに高密度の点群地図が作成することができます．ただし点群データの全体的な整合性を考えてマッピン

グされた結果とはならないため，確実に正確な地図が得られる保証がないことには留意しなければなりません．

本章で紹介した方法は，ポーズグラフを用いたグラフベース SLAM になります．ポーズグラフを用いる場合，LiDAR の点群データは最適化対象にならなくなるため，マッピングの精度は通常のグラフベース SLAM と比較すると低下してしまいます．ただし，通常グラフベース SLAM では，メモリコストの問題があるため，LiDAR が計測した点群すべてを最適化対象にはせず，特徴点やランドマークを LiDAR の点群から検出し，これらの位置を最適化対象に加えます．そのため，特徴点やランドマークの検出性能，またそれらの対応付けの性能がマッピングの精度にも影響することになります．近年の 3D LiDAR は観測距離も長く，点群の密度も高いため，単純に 2 つの計測データを照合するだけでも，高い精度で相対位置を知ることができます．そのため，このような前提であれば，ポーズグラフの最適化だけでも十分な精度の地図を得られることが多いです．ただし，文献 [] に見られるように，地図全体を最適化するというプロセスを踏むと，より地図の精度を向上させることもできるため，ポーズグラフの最適化だけで十分な精度が得られるかどうかはケースバイケースであるといえます．

本章で述べた方法では，ループ検知を行うために，単純に近隣のキーフレーム間でスキャンマッチングを行うという方法を用いています．3.5 でも述べましたが，スキャンマッチングが正しく機能する前提の 1 つに初期姿勢がある程度正確に取得できてることが挙げられます．ただしオドメトリでは，移動に対する累積（ドリフト）誤差が無視できないため，SLAM 実行中の移動量が長くなるにつれ，ループ検知を行うためのスキャンマッチングの初期姿勢の精度が低下してしまいます．そのため，単純にスキャンマッチングを用いてループ検知を行うだけでは，大きなループを閉じることができない場合もあります．そのような場合には，例えば LiDAR の計測点群から特徴点や特徴量を抽出し，複数の点群から類似性の高い点群を検出するという方法を用いなければなりません．ただしこのような方法は，スキャンマッチングでループを検出するより，一般的に精度が低下してしまうことが多いです．そのため，ポーズグラフの最適化時におけるアウトライアの検出なども考慮する必要が発生することもあります．

なおスキャンマッチングを用いてループ検出を行う場合だけでも，正しくループが閉じることができてれば，オドメトリによる累積誤差が補正できていることを意味します．そのため，ループが閉じやすいような経路を考えてデータを取得することで，地図作成を成功させるということも実用的にはあります．ただしループが大きすぎる場合は性能の限界があるため，違う方法の導入を検討しなければなりません．

参考文献

[1] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1322–1328, 1999.

[2] N. Akai. Reliable Monte Carlo localization for mobile robots. *Journal of Field Robotics*, 40(3):595–613, 2023.

[3] W. Xu and F. Zhang. FAST-LIO: A fast, robust LiDAR–inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robotics and Automation Letters*, 6(2):3317–3324, 2021.

[4] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[5] J. Borenstein, H.R. Everett, L. Feng, and D. Wehe. Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems*, 14(4):229–340, 1997.

[6] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, 2004.

[7] J. Zhang and S. Singh. LOAM: lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, 2014.

[8] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2017.

[9] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and Rus D. LIO–SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142. IEEE, 2020.

[10] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang. FAST-LIO2: Fast direct LiDAR–inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022.

[11] K. Koide, M. Yokozuka, S. Oishi, and A. Banno. GLIM: 3D range-inertial localization and mapping with GPU-accelerated scan matching factors. *Robotics and Autonomous Systems*, 179(2):104750, 2024.

[12] K. Chen, R. Nemiroff, and B. T. Lopez. Direct lidar-inertial odometry: Lightweight LIO with continuous-time motion correction. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3983–3989, 2023.

[13] B. T. Lopez. A contracting hierarchical observer for pose-inertial fusion. *arXiv:2303.02777*, 2023.

[14] A. Segal, D. Hähnel, and S. Thrun. Generalized–ICP. In Jeff Trinkle, Yoky Matsuoka, and José A. Castellanos, editors, *Robotics: Science and Systems*. The MIT Press, 2009.