

グラフベース SLAM 入門

杉浦 圭祐

慶應義塾大学理工学部情報工学科 松谷研究室

May 27, 2020

- ① イン트로ダクション
 - SLAM の概要
 - SLAM の大まかな流れ
 - ループ閉じ込み
 - 逐次処理と一括処理
- ② グラフベース SLAM とは
- ③ グラフベース SLAM の定式化
 - 完全 SLAM 問題
 - ポーズグラフのエッジ
 - ポーズ調整
 - まとめ
- ④ ポーズグラフの構築
 - 逐次的な SLAM による構築
 - ループ検出による構築
 - まとめ

5 ポーズ調整の解法

- ポーズ調整の概要
- ポーズ調整の入出力
- ポーズ調整のアルゴリズムの導出

6 ポーズ調整のまとめ

- ポーズ調整の入出力
- ポーズ調整のアルゴリズム
- 1/4: 方程式の係数の計算
- 2/4: 方程式を解く
- 3/4: 解の更新
- 4/4: 収束判定
- ポーズ調整のためのライブラリ

7 細かな話題

- ループ辺の必要性
- ヤコビ行列の計算例

目次

- レーベンバーグ・マーカート法の利用
- 外れ値への対処

8 まとめ

目次

- ① イン트로ダクション
- ② グラフベース SLAM とは
- ③ グラフベース SLAM の定式化
- ④ ポーズグラフの構築
- ⑤ ポーズ調整の解法
- ⑥ ポーズ調整のまとめ
- ⑦ 細かな話題
- ⑧ まとめ

① イン트로ダクション

- SLAM の概要
- SLAM の大まかな流れ
- ループ閉じ込み
- 逐次処理と一括処理

- SLAM: Simultaneous Localization And Mapping
 - 地図構築と自己位置推定の同時実行
- SLAM の入力と出力
 - **入力**: 制御 / センサデータ
 - **出力**: ロボットの姿勢 / 地図
- ここで想定する SLAM
 - 2 次元
 - グラフベース
 - レーザスキャナ (**LiDAR**) を使用
 - 地図の形式については考慮しない

- SLAM の入力と出力

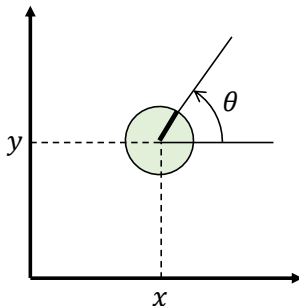
- **入力**: 制御 / センサデータ
- **出力**: ロボットの姿勢 / 地図

- 記号の整理

- 時刻 t ($t \geq 0$, t は整数)
- ロボットの姿勢 $x_t = [\xi_t^x, \xi_t^y, \xi_t^\theta]^\top$ (位置 x, y と回転角 θ)
- 制御 $u_t = [\delta_t^x, \delta_t^y, \delta_t^\theta]^\top$ (相対姿勢)
- センサデータ $z_t = \{z_t^i\}$, $z_t^i = [r_t^i, \theta_t^i]^\top$
センサ中心から障害物までの, 距離と方向の集合 (**スキャンデータ**)
- 地図 m

SLAM の概要

- ロボットの姿勢 $x_t = [\xi_t^x, \xi_t^y, \xi_t^\theta]^\top$
 - ワールド座標系で表現
 - 位置の成分 ξ^x, ξ^y と回転角の成分 ξ^θ
 - ξ^θ はヨー角 (z 軸まわりの回転)

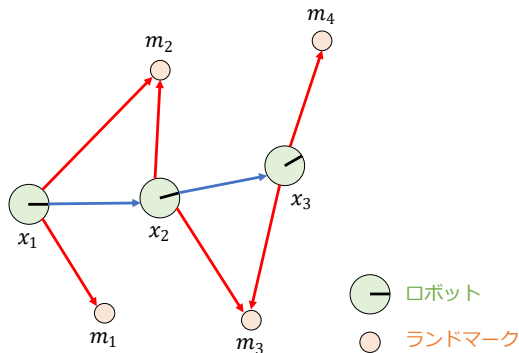


① イン트로ダクション

- SLAM の概要
- SLAM の大まかな流れ
- ループ閉じ込み
- 逐次処理と一括処理

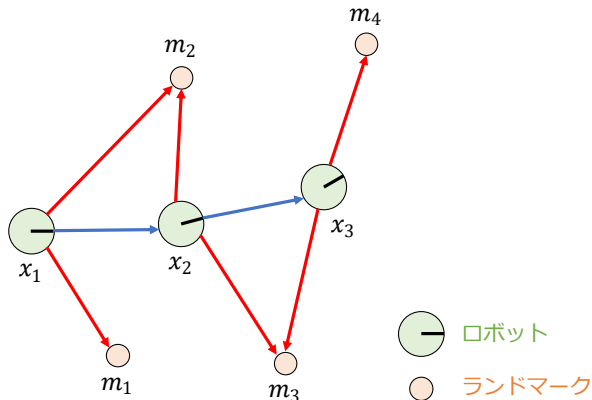
SLAM の流れ

- ロボットは、各時刻においてランドマークを観測 [16]
 - ランドマーク: 位置の目印 (特徴的なもの)
 - ロボット中心からランドマークまでの, 距離と方向を計測
 - ロボットは、各ランドマークを区別できるとする (簡単のため)



SLAM の流れ

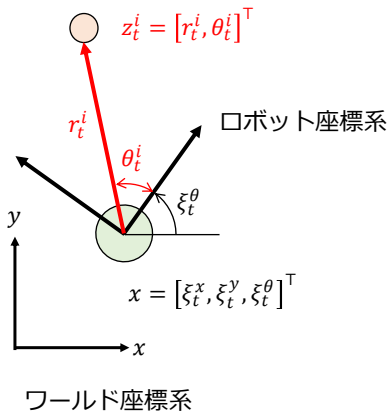
- ロボットは、各時刻においてランドマークを観測 [16]
 - 時刻 $t = 1$ で、ロボットはランドマーク m_1 と m_2 を観測
 - 計測されたランドマークを合わせて、地図を構築



- ロボットは、各時刻においてランドマークを観測 [16]
 - ランドマークは、ロボット座標系で表現される
 - ⇒ ロボットの姿勢を用いて、ワールド座標系へ変換
 - ⇒ 地図にランドマークを追加して拡張
- 地図を構築するには、ロボットの姿勢が必要
 - ロボットの姿勢 x_t は、逐次的に更新される
 - 以前の姿勢 x_{t-1} に、制御 u_t を適用させて、 x_t を得る
 - 自己位置推定の基本となる処理

SLAM の流れ

- ロボットは、各時刻においてランドマークを観測 [16]
 - 時刻 t で、 j 番目のランドマーク m_j を観測
 - 計測 z_t の i 番目 $z_t^i = [r_t^i, \theta_t^i]^\top$ が、 m_j に対応しているとする

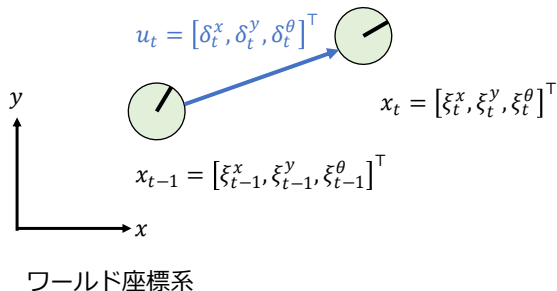


- ロボットは、各時刻においてランドマークを観測 [16]
 - $z_t^i = [r_t^i, \theta_t^i]^\top$ は、ロボット座標系で表現 (極座標)
 - r_t^i, θ_t^i は、ロボット中心からランドマークまでの距離と角度
- ワールド座標系 $p_t^i = [p_t^{i,x}, p_t^{i,y}]^\top$ に変換する式

$$\begin{bmatrix} p_t^{i,x} \\ p_t^{i,y} \end{bmatrix} = \begin{bmatrix} \xi_t^x \\ \xi_t^y \end{bmatrix} + \begin{bmatrix} r_t^i \cdot \cos(\xi_t^\theta + \theta_t^i) \\ r_t^i \cdot \sin(\xi_t^\theta + \theta_t^i) \end{bmatrix} \quad (1)$$

- ロボットは、各時刻においてランドマークを観測 [16]
 - ランドマークは、ロボット座標系で表現される
 - ⇒ ロボットの姿勢を用いて、ワールド座標系へ変換
 - ⇒ 地図にランドマークを追加して拡張
- 地図を構築するには、ロボットの姿勢が必要
 - ロボットの姿勢 x_t は、逐次的に更新される
 - 以前の姿勢 x_{t-1} に、制御 u_t を適用させて、 x_t を得る
 - 自己位置推定の基本となる処理

- ロボットの姿勢の更新 [12] [16]
 - 以前の姿勢 x_{t-1} に, 制御 u_t を適用させて, x_t を得る
 - 制御 $u_t = [\delta_t^x, \delta_t^y, \delta_t^\theta]^\top$ は, x_{t-1} と x_t の**相対姿勢**とする
 - $u_t = [\delta_t^x, \delta_t^y, \delta_t^\theta]^\top$ は, **ロボット座標系**で表現



- ロボットの姿勢の更新 [16]

- x_{t-1} と u_t から, 新たな姿勢 x_t を求める式

$$\begin{bmatrix} \xi_t^x \\ \xi_t^y \\ \xi_t^\theta \end{bmatrix} = \begin{bmatrix} \cos \xi_{t-1}^\theta & -\sin \xi_{t-1}^\theta & 0 \\ \sin \xi_{t-1}^\theta & \cos \xi_{t-1}^\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta_t^x \\ \delta_t^y \\ \delta_t^\theta \end{bmatrix} + \begin{bmatrix} \xi_{t-1}^x \\ \xi_{t-1}^y \\ \xi_{t-1}^\theta \end{bmatrix} \quad (2)$$

- **Compounding 演算子 \oplus** を用いて, $x_t = x_{t-1} \oplus u_t$ と表現

- x_{t-1} と x_t から, 制御 u_t (相対姿勢) を求める式

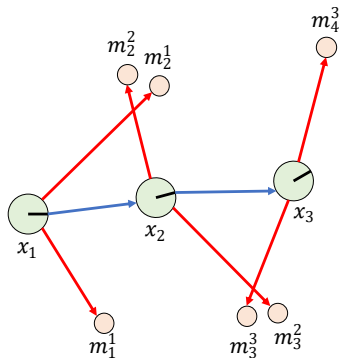
$$\begin{bmatrix} \delta_t^x \\ \delta_t^y \\ \delta_t^\theta \end{bmatrix} = \begin{bmatrix} \cos \xi_{t-1}^\theta & \sin \xi_{t-1}^\theta & 0 \\ -\sin \xi_{t-1}^\theta & \cos \xi_{t-1}^\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \xi_t^x - \xi_{t-1}^x \\ \xi_t^y - \xi_{t-1}^y \\ \xi_t^\theta - \xi_{t-1}^\theta \end{bmatrix} \quad (3)$$

- **Inverse Compounding 演算子 \ominus** を用いて, $u_t = x_t \ominus x_{t-1}$ と表現

- $u_t = x_t \ominus x_{t-1}$ であるとき, $x_t = x_{t-1} \oplus u_t$ が成り立っている

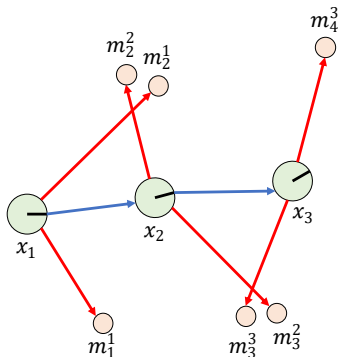
- 誤差の影響を常にする [16]

- m_j^i : 位置 x_i で観測した j 番目のランドマーク (ワールド座標系)
- 同一のランドマークを観測していても、位置がずれる ($m_2^1 \neq m_2^2$)
⇐ 制御 u_1 と計測 z_1, z_2 に誤差があるため

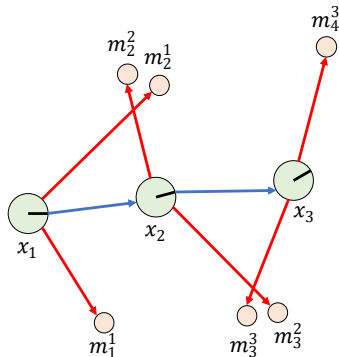


- 誤差の影響を常に受ける [16]

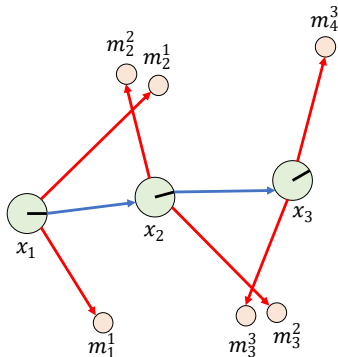
- m_2^2 は 2 番目のランドマークで, m_2^1 と重なるべき
- m_3^3 は 3 番目のランドマークで, m_3^2 と重なるべき
- ランドマーク同士が重なり合うように, ロボットの位置 x_2, x_3 を修正



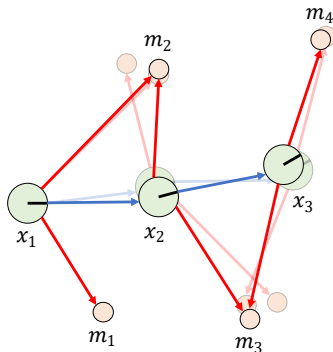
- 誤差の影響を常に受ける [16]
 - ロボットの姿勢が x_2 であるとき, m_1^1 と m_2^1 は地図に取り込まれている
 - m_2^2 が m_2^1 と同じ位置になるように, x_2 を修正
- ⇒ 修正した姿勢 x_2 の下で, ランドマーク m_2^2, m_3^2 を地図上に配置



- 誤差の影響を常に受ける [16]
 - ロボットの姿勢が x_3 であるとき, m_3^2 は地図に取り込まれている
 - m_3^3 が m_3^2 と同じ位置になるように, x_3 を修正
- ⇒ 修正した姿勢 x_3 の下で, ランドマーク m_3^3, m_4^3 を地図上に配置



- 誤差の影響を常に受ける [16]
 - ランドマークと地図との重ね合わせにより、精度の良い姿勢を得る
 - この重ね合わせ処理を、**スキャンマッチング**という
 - スキャンマッチングを行うためには、地図が必要



- 自己位置推定と地図構築のステップ

- 1 以前の姿勢 x_{t-1} に制御 u_t を適用させ, 新たな姿勢 x'_t を得る

- 2 スキャンマッチングにより, 改良した姿勢 x_t を得る

- ⇐ ランドマークと地図との重ね合わせ処理

- 3 改良した姿勢 x_t を使って, ランドマークを地図に追加

- 制御 u_t を使わなくても, 新たな姿勢 x_t は計算できる

- ⇐ 以前の姿勢 x_{t-1} を初期値としたスキャンマッチングを実行

- 自己位置推定を行うためには, 地図が必要

- ⇔ 地図を構築するには, ロボットの姿勢 (自己位置) が必要

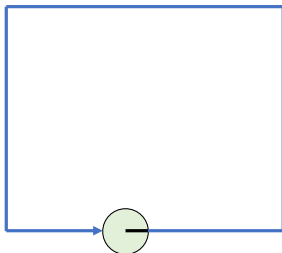
- 自己位置推定と地図構築の間には、相互依存関係がある
 - 誤差が互いに影響し合う \Rightarrow 誤差が累積する
 - 自己位置推定に含まれる誤差 \Rightarrow 地図 (ランドマーク位置) の誤差
 - 地図 (ランドマーク位置) の誤差 \Rightarrow 自己位置推定の誤差
- 上記のステップだけでは、誤差が累積してしまう
 - ループ閉じ込みにより解決
 - 文献 [16] の 10 章を基に説明

① イン트로ダクション

- SLAM の概要
- SLAM の大まかな流れ
- ループ閉じ込み
- 逐次処理と一括処理

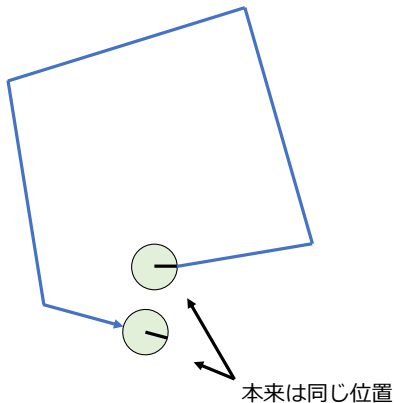
ループ閉じ込み

- ロボットが建物内を一周して, 同じ場所に戻ってくる [16]
 - 誤差がなければ, ロボットの軌跡はループ状になる (ループが閉じる)



ループ閉じ込み

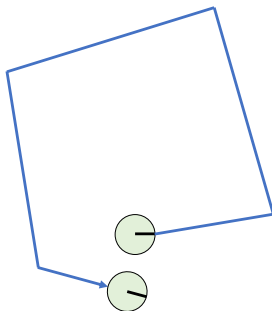
- ロボットが建物内を一周して、同じ場所に戻ってくる [16]
 - 実際には累積誤差の影響で、同じ場所には戻らない (ループが閉じない)
 - 地図が歪む



ループ閉じ込み

- 地図の歪みの影響 [16]

- 地図上では通路になっているが、実際には壁や障害物があって通れない
- 同一の通路が 2 箇所に見える (同一の場所が地図上に複数現れる)
- 本来は交差するはずの通路が交わらない / 偽の交差点ができる
- 目的地までたどり着けない (ナビゲーションに使えない)



ループ閉じ込み

- ループ閉じ込みの手順 [16]

- 1 ループ検出 (Loop Detection)

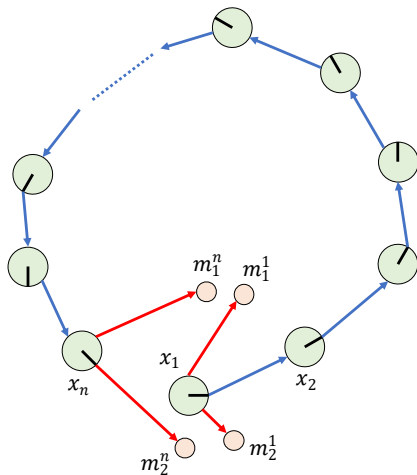
ロボットが同じ場所に戻ってきたことを検出

- 2 ロボットの軌跡と地図 (ランドマーク位置) の修正

ループが閉じるように, 軌跡と地図の双方を修正 (辻褄を合わせる)

ループ閉じ込み

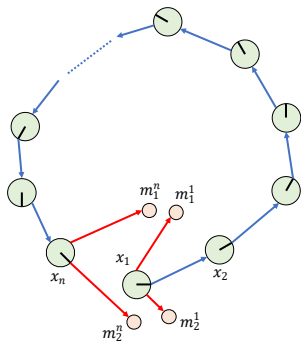
- ループ検出 [16]
 - x_n において, x_1 のときと同じランドマークを複数観測した



ループ閉じ込み

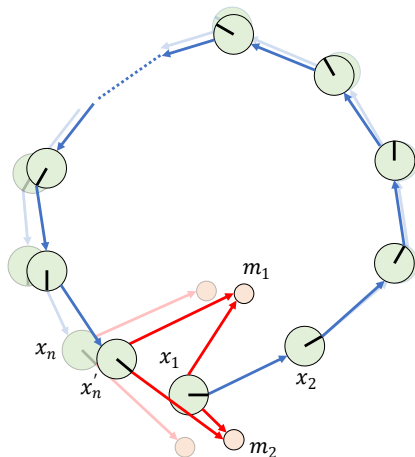
- ループ検出 [16]

- x_n において, x_1 のときと同じランドマークを複数観測した
- x_n は累積誤差の影響を受けている
⇒ ランドマークの位置が重ならない ($m_1^1 \neq m_1^n, m_2^1 \neq m_2^n$)
- ランドマークが重なり合うように, 位置 x_n を修正して, x'_n を得る



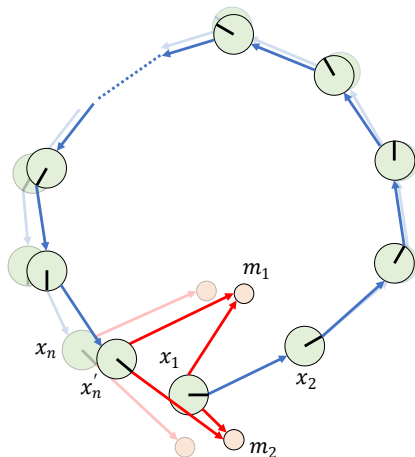
ループ閉じ込み

- ロボットの軌跡と地図 (ランドマーク位置) の修正 [16]
 - 修正した x'_n に合わせて, ロボットの軌跡と地図を修正



ループ閉じ込み

- ロボットの軌跡と地図 (ランドマーク位置) の修正 [16]
 - 軌跡と地図全体が修正される \Rightarrow 累積誤差が一気に解消される



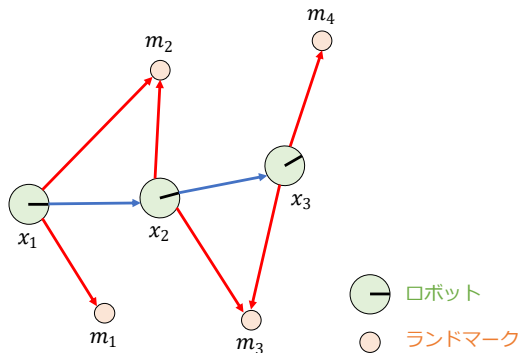
① イン트로ダクション

- SLAM の概要
- SLAM の大まかな流れ
- ループ閉じ込み
- 逐次処理と一括処理

逐次処理と一括処理

- 逐次処理 [16]

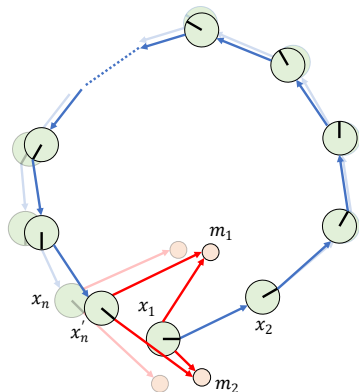
- リアルタイム処理 (次のセンサデータが来るまでに完了すべき)
- 制御の反映と, スキャンマッチングによる姿勢の更新
- 最新の (少量の) スキャンデータを繋ぎ合わせて地図を拡張
- 局所的には正確な地図が作成できる



逐次処理と一括処理

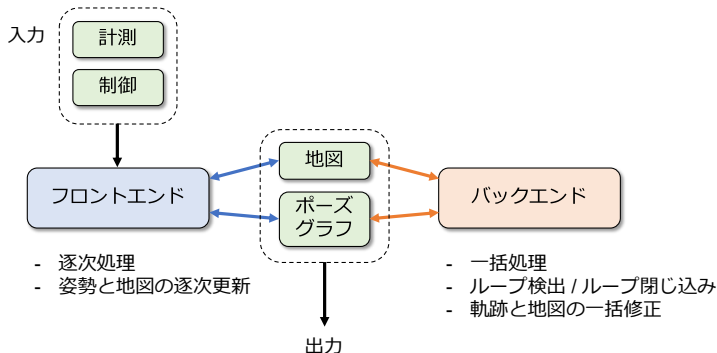
- 一括処理 [16]

- オフライン処理 / 最適化問題を解く
- 全てのセンサデータを使って、軌跡と地図を修正
- 整合性の取れた、正確な地図を作成できる



逐次処理と一括処理

- 逐次処理と一括処理の組み合わせ [16]
 - マルチスレッドによる実行 (役割分担)
 - フロントエンド: 逐次処理 (ローカルな処理, グラフの構築)
 - バックエンド: 一括処理 (グローバルな処理, グラフの最適化)



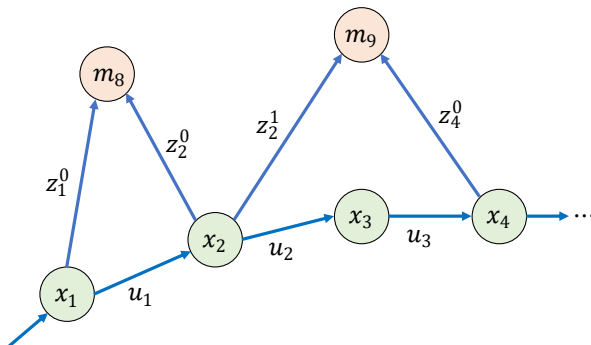
- SLAM: 局所的な SLAM と大域的な SLAM から構成
- 局所的な SLAM
 - 姿勢の更新: 制御の反映と, スキャンマッチング
 - 地図の更新: 現在の姿勢の下で, スキャンデータを地図に追加
 - スキャンマッチング: 地図とスキャンデータ (計測) の重ね合わせ
- 大域的な SLAM
 - ループ検出
 - ループ閉じ込み
 - ロボットの軌跡と地図の一括修正

目次

- ① イントロダクション
- ② **グラフベース SLAM とは**
- ③ グラフベース SLAM の定式化
- ④ ポーズグラフの構築
- ⑤ ポーズ調整の解法
- ⑥ ポーズ調整のまとめ
- ⑦ 細かな話題
- ⑧ まとめ

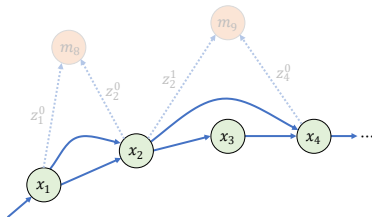
グラフベース SLAM とは

- グラフを構築する SLAM [12, 5, 16, 17]
 - **ノード**: 各時刻におけるロボットの姿勢 / ランドマークの位置
 - **エッジ**: 計測結果 (ノード間の**相対姿勢**として得られる)



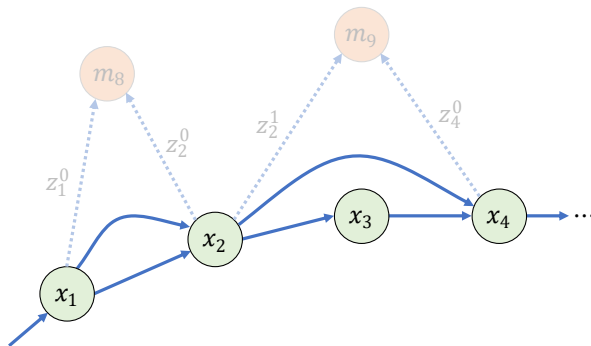
グラフベース SLAM とは

- グラフを構築する SLAM [12, 5, 16, 17]
 - 先程のグラフは、姿勢とランドマーク位置 (地図) をノードとしてもつ
 - ⇒ 地図を構成するランドマーク数は、大規模になり得る (数万から数億)
 - ⇒ ノードの数は、最適化問題における変数の個数に直結
 - ⇒ ノードの数が多く、計算量が膨大になる
 - グラフを簡略化し、ロボットの姿勢のみをノードとして持たせる
 - ⇒ これを **ポーズグラフ** という (計算量を削減できる)



グラフベース SLAM とは

- ポーズグラフ [16]
 - ノード: 各時刻におけるロボットの姿勢 x_t
 - エッジ: ノード間の拘束
 - 拘束には, 計測やループに関する情報が埋め込まれている



グラフベース SLAM とは

- ポーズグラフによる SLAM の流れ

- 1 計測やループ検出などから得た情報を基に, ポーズグラフを構築
- 2 ポーズグラフを, 最適化問題に変換
- 3 最適化問題を解き, ポーズグラフを最適化
- 4 最適化されたポーズグラフを使って, ロボットの軌跡と地図を修正

- 課題

- グラフベース SLAM の定式化
- ポーズグラフを構築する方法
- 最適化問題を構成する方法
- 最適化問題を解く方法

目次

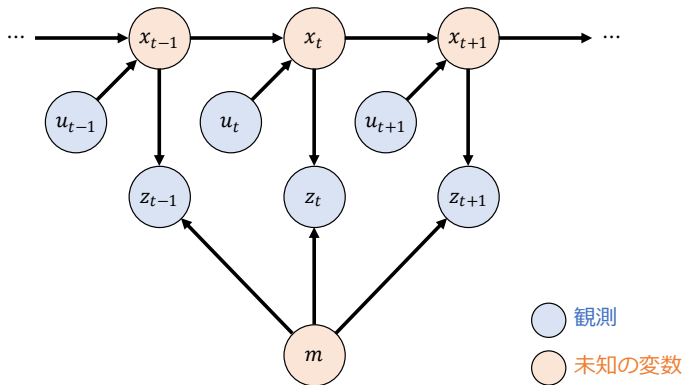
- ① イントロダクション
- ② グラフベース SLAM とは
- ③ グラフベース SLAM の定式化**
- ④ ポーズグラフの構築
- ⑤ ポーズ調整の解法
- ⑥ ポーズ調整のまとめ
- ⑦ 細かな話題
- ⑧ まとめ

③ グラフベース SLAM の定式化

- 完全 SLAM 問題
- ポーズグラフのエッジ
- ポーズ調整
- まとめ

グラフベース SLAM の定式化

- グラフベース SLAM は、**完全 SLAM** 問題を解く [5, 16]
 - ロボットの**軌跡** $x_{1:t}$ と**地図** m を計算 (軌跡: 全時刻における姿勢)
 - 過去から現在に至るまでの、**全ての**制御 $u_{0:t}$, 計測 $z_{0:t}$ を利用



グラフベース SLAM の定式化

- グラフベース SLAM は, 完全 SLAM 問題を解く
- 完全 SLAM 問題 [17]
 - 下記の確率密度を最大化する, ロボットの軌跡 $x_{1:t}$ と地図 m を計算
 - 過去から現在に至るまでの, 全ての制御 $u_{0:t}$, 計測 $z_{0:t}$ を利用

$$x_{0:t}^*, m^* = \arg \max_{x_{0:t}, m} p(x_{0:t}, m \mid \hat{x}_0, u_{0:t}, z_{0:t}) \quad (4)$$

- x_0, \dots, x_t の相対的な位置関係だけが分かる
 - $\Rightarrow x_0, \dots, x_t$ は, 剛体変換に対して不変
 - $\Rightarrow x_{0:t}$ が一意に定まるためには, 姿勢のどれかを固定する必要
 - \Rightarrow 初期姿勢 x_0 を, \hat{x}_0 に固定
- 初期姿勢 x_0 は \hat{x}_0 から殆ど変化しないが, 便宜上変数として扱う

グラフベース SLAM の定式化

- 完全 SLAM 問題の分割 [17]

- $x_{0:t}^*, m^* = p(x_{0:t}, m \mid \hat{x}_0, u_{0:t}, z_{0:t})$
- 地図 m は大規模になる (数万から数億次元)
⇒ 上記の問題を直接解くのは、計算量の観点から困難
- 完全 SLAM 問題を、軌跡の推定と地図の推定に分割

$$p(x_{0:t}, m \mid \hat{x}_0, u_{0:t}, z_{0:t}) = \underbrace{p(x_{0:t} \mid \hat{x}_0, u_{0:t}, z_{0:t})}_{\text{軌跡の推定}} \underbrace{p(m \mid x_{0:t}, z_{0:t})}_{\text{地図の推定}} \quad (5)$$

- 軌跡の推定問題を、**ポーズ調整** (Pose Adjustment) という

$$x_{0:t}^* = \arg \max_{x_{0:t}} p(x_{0:t} \mid \hat{x}_0, u_{0:t}, z_{0:t}) \quad (6)$$

- **ポーズ調整**を行うことで、ポーズグラフを最適化

グラフベース SLAM の定式化

- 完全 SLAM 問題の分割 [17]
 - 完全 SLAM 問題を, 軌跡の推定と地図の推定に分割
 - 軌跡の推定 (ポーズ調整)

$$x_{0:t}^* = \arg \max_{x_{0:t}} p(x_{0:t} \mid \hat{x}_0, u_{0:t}, z_{0:t})$$

- 地図の推定

$$m^* = \arg \max_m p(m \mid x_{0:t}, z_{0:t}) \quad (7)$$

- 地図の推定の際は, ポーズ調整で得た軌跡 $x_{0:t}^*$ を確定値として扱う
⇒ 地図は, 確定値 $x_{0:t}^*$ に対して決定論的に計算される

$$m^* \simeq m(x_{0:t}^*, z_{0:t}) \quad (8)$$

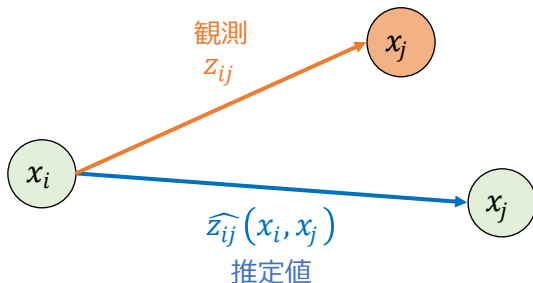
- 地図の推定は考えないことにして, ポーズ調整に焦点を当てる

③ グラフベース SLAM の定式化

- 完全 SLAM 問題
- ポーズグラフのエッジ
- ポーズ調整
- まとめ

グラフベース SLAM の定式化

- ポーズグラフ [5]
 - ポーズグラフのエッジは、ノード間の拘束を表現する
 - ノードは、各時刻でのロボットの姿勢である
 - 具体的に、エッジはどのような情報を持つか？
 - 時刻 i, j におけるノード x_i, x_j を結ぶエッジを考える



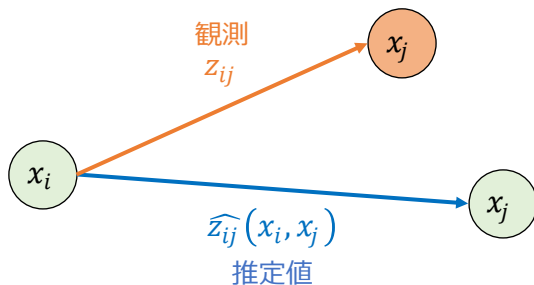
グラフベース SLAM の定式化

- ポーズグラフのエッジ [5]

- 時刻 i, j におけるノード x_i, x_j を結ぶエッジを考える

- 観測によって, 2 ノード間の相対姿勢 z_{ij} が得られている

⇐ 観測とは, 計測によるスキャンマッチングや, ループ検出を指す



グラフベース SLAM の定式化

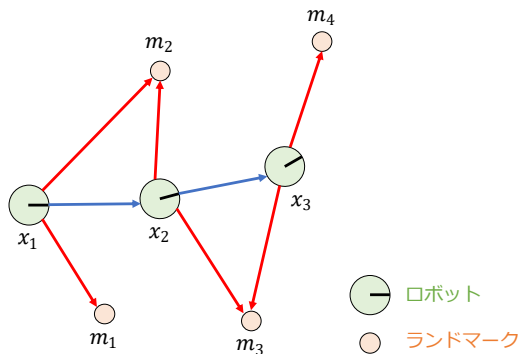
- ポーズグラフのエッジ [16]

- 例: 逐次処理 (制御の反映と, スキャンマッチング)

- ⇒ 以前の姿勢 x_{t-1} に制御 u_t を反映

- ⇒ 計測 z_t と地図 m を重ね合わせて, 新たな姿勢 x_t を計算

- ⇒ 隣接する時刻 $t-1, t$ における相対姿勢 $z_{t-1,t}$ が得られる



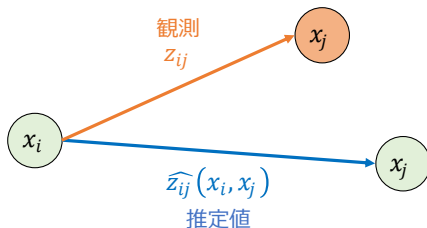
グラフベース SLAM の定式化

- ポーズグラフのエッジ [5]

- 2 ノード間の相対姿勢は, 姿勢 x_i, x_j から次のようにも計算できる
- これを $\hat{z}_{ij}(x_i, x_j)$ とする

$$\hat{z}_{ij}(x_i, x_j) = x_j \ominus x_i = \begin{bmatrix} \cos \xi_i^\theta & \sin \xi_i^\theta & 0 \\ -\sin \xi_i^\theta & \cos \xi_i^\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \xi_j^x - \xi_i^x \\ \xi_j^y - \xi_i^y \\ \xi_j^\theta - \xi_i^\theta \end{bmatrix} \quad (9)$$

- 但し, $x_i = [\xi_i^x, \xi_i^y, \xi_i^\theta]^\top$, $x_j = [\xi_j^x, \xi_j^y, \xi_j^\theta]^\top$ である



グラフベース SLAM の定式化

- ポーズグラフのエッジ [5]
 - 観測により得られた相対姿勢 z_{ij} は, 定数値
 - 相対姿勢 $\hat{z}_{ij}(x_i, x_j)$ は, 変数 x_i, x_j から計算される
 $\Rightarrow \hat{z}_{ij}(x_i, x_j)$ は変数を含んでおり, 最適化の前後で変化
 - 次の残差の最小化を考える

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j) \quad (10)$$

- 残差は, x_i, x_j についての関数 (観測で得た z_{ij} は定数)

グラフベース SLAM の定式化

- ポーズグラフのエッジ [17]
 - 次の残差の最小化を考える

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$$

- 誤差がなければ, 残差 $e_{ij}(x_i, x_j)$ は 0 になる
 - ⇒ 実際には, 誤差の影響で 0 にならない
- 局所的には, あるエッジの残差を 0 にすればよい
 - ⇒ ノードを介して, 全てのエッジ (残差) が互いに影響し合う
 - ⇒ 大域的に (全てのエッジについて), 残差を考慮する必要

グラフベース SLAM の定式化

- 残差のモデル化 [5, 17]

- 残差 $e_{ij}(x_i, x_j)$ の確率分布を, ガウス分布で表現

$$p(e_{ij}) = \mathcal{N}(e_{ij} \mid 0, \Sigma_{ij}) \quad (11)$$

$$\propto \exp \left\{ -\frac{1}{2} e_{ij}(x_i, x_j)^\top \Sigma_{ij}^{-1} e_{ij}(x_i, x_j) \right\} \quad (12)$$

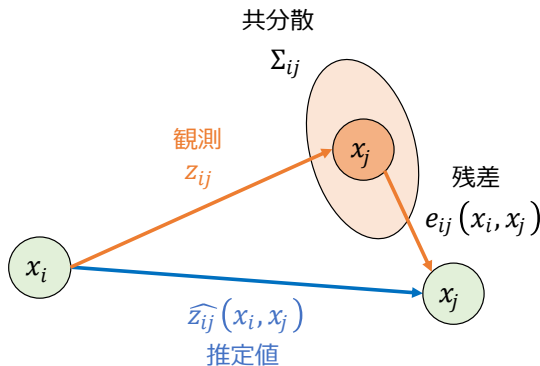
- 平均 0, 共分散 Σ_{ij} のガウス分布
- 共分散 Σ_{ij} は, 何らかの方法で決める必要
- 観測結果 (z_{ij}) に誤差が少なく信頼できるなら, Σ_{ij} を小さく設定
 - ⇒ 残差 e_{ij} が大きくなると, $p(e_{ij})$ は急激に減少
 - ⇒ ノード x_i, x_j は, 観測結果 z_{ij} に従おうとする
- $p(e_{ij})$ は, ノード i, j を結ぶエッジの整合性を表す

グラフベース SLAM の定式化

- 残差のモデル化 [5]

- 残差 $e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$ の確率分布

$$p(e_{ij}) \propto \exp \left\{ -\frac{1}{2} e_{ij}(x_i, x_j)^\top \Sigma_{ij}^{-1} e_{ij}(x_i, x_j) \right\}$$



グラフベース SLAM の定式化

- まとめ

- 各エッジに対して、次の**残差**を最小化

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$$

- z_{ij} : 観測によって得られた, ノード i, j 間の相対姿勢 (**定数**)
- $\hat{z}_{ij}(x_i, x_j)$: 最適化によって推定される相対姿勢 (x_i, x_j は**変数**)
- 残差の最小化は, 次のガウス分布の**最大化**に相当

$$p(e_{ij}) \propto \exp \left\{ -\frac{1}{2} e_{ij}(x_i, x_j)^\top \Sigma_{ij}^{-1} e_{ij}(x_i, x_j) \right\}$$

- 各エッジは, 観測による相対姿勢 z_{ij} と, 共分散 Σ_{ij}^{-1} を保持する

③ グラフベース SLAM の定式化

- 完全 SLAM 問題
- ポーズグラフのエッジ
- ポーズ調整
- まとめ

グラフベース SLAM の定式化

- ポーズ調整 [17]

- 以下の関数 $f(x_{0:t})$ を, 軌跡 $x_{0:t}$ (全ノードの姿勢) について **最大化**

$$f(x_{0:t}) = \prod_{(i,j) \in \mathcal{C}} p(e_{ij}) \quad (12)$$

$$\propto \prod_{(i,j) \in \mathcal{C}} \exp \left\{ -\frac{1}{2} e_{ij}(x_i, x_j)^\top \Sigma_{ij}^{-1} e_{ij}(x_i, x_j) \right\} \quad (13)$$

- 関数 $f(x_{0:t})$ は, 全エッジに対する確率密度 $p(e_{ij})$ の積
- \mathcal{C} は, ポーズグラフ内の全てのエッジ (i, j) の集合
⇐ 各エッジを, 両端のノードのインデックス (i, j) で表現

グラフベース SLAM の定式化

- 初期姿勢 x_0 の固定 [17]
 - 以下の関数を, 全姿勢 $x_{0:t}$ について最大化

$$f(x_{0:t}) \propto \prod_{(i,j) \in \mathcal{C}} \exp \left\{ -\frac{1}{2} e_{ij}(x_i, x_j)^\top \Sigma_{ij}^{-1} e_{ij}(x_i, x_j) \right\}$$

- 初期姿勢 x_0 を, \hat{x}_0 に固定する必要があった
- 以下の $p_0(x_0)$ を, 平均が \hat{x}_0 で, 分散が非常に小さなガウス分布とする

$$p_0(x_0) \propto \exp \left\{ -\frac{1}{2} (x_0 - \hat{x}_0)^\top \Sigma_0^{-1} (x_0 - \hat{x}_0) \right\} \quad (14)$$

- 共分散 Σ_0 は対角行列で, 各対角要素は非常に小さな値をもつ
 - $\Rightarrow \Sigma_0 = \varepsilon I, \varepsilon \ll 1$ (I は単位行列)
 - $\Rightarrow x_0$ の x, y, θ 成分に加わる誤差は, 互いに独立とする

グラフベース SLAM の定式化

- 初期姿勢 x_0 の固定 [17]
 - 先程の関数 $f(x_{0:t})$ に, $p_0(x_0)$ を追加

$$\begin{aligned} & f(x_{0:t}) \\ \propto & p_0(x_0) \prod_{(i,j) \in \mathcal{C}} p(e_{ij}) \end{aligned} \quad (15)$$

$$\propto p_0(x_0) \prod_{(i,j) \in \mathcal{C}} \exp \left\{ -\frac{1}{2} e_{ij}(x_i, x_j)^\top \Sigma_{ij}^{-1} e_{ij}(x_i, x_j) \right\} \quad (16)$$

- 初期姿勢 x_0 は, \hat{x}_0 に固定する必要がある
 - \Leftarrow 軌跡 $x_{0:t}$ は, x, y, θ 方向に平行移動させても不変
- 上記の $f(x_{0:t})$ は, x_0 が \hat{x}_0 から少しでもずれると, 急激に減少
 - $\Rightarrow x_0$ を \hat{x}_0 に固定しながら, $x_{1:t}$ を調整できる

- 情報行列 Ω による表記 [17]
 - 共分散行列の逆行列 Σ^{-1} を, 精度行列や情報行列 Ω という
 - これ以降, **情報行列** Ω を用いて表記
 - x_0 の確率分布 ($\Omega_0 = \Sigma_0^{-1}$, $\Omega_0 = \varepsilon^{-1}I$, $\varepsilon \ll 1$)

$$p_0(x_0) \propto \exp \left\{ -\frac{1}{2} (x_0 - \hat{x}_0)^\top \Omega_0 (x_0 - \hat{x}_0) \right\} \quad (17)$$

- Ω_0 の各対角成分は, 十分に大きな値
- グラフベース SLAM で解く問題 ($\Omega_{ij} = \Sigma_{ij}^{-1}$)

$$\begin{aligned} f(x_{0:t}) &\propto p_0(x_0) \prod_{(i,j) \in \mathcal{C}} p(e_{ij}) \\ &\propto p_0(x_0) \prod_{(i,j) \in \mathcal{C}} \exp \left\{ -\frac{1}{2} e_{ij}^\top \Omega_{ij} e_{ij} \right\} \end{aligned} \quad (18)$$

- 負の対数の最大化 [17]

- $f(x_{0:t})$ は指数関数であるから, 単調増加
⇒ 対数 $\ln f(x_{0:t})$ を最大化しても, 同じ解 $x_{0:t}$ が得られる

$$\begin{aligned} & \ln f(x_{0:t}) \\ &= \eta + \ln p_0(x_0) + \sum_{(i,j) \in \mathcal{C}} \ln p(e_{ij}) \\ &= \eta - \frac{1}{2} (x_0 - \hat{x}_0)^\top \Omega_0 (x_0 - \hat{x}_0) - \\ & \quad \sum_{(i,j) \in \mathcal{C}} \frac{1}{2} e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j) \end{aligned}$$

- 関数 $f(x_{0:t})$ の最大化は, $\ln f(x_{0:t})$ の最大化と等価
⇒ $\ln f(x_{0:t})$ の定数 η を除去し, 符号を反転させて, 次の結果を得る

グラフベース SLAM の定式化

- ポーズ調整 [5, 17]
 - 結局, $f(x_{0:t})$ の最大化は, 次の関数 $F(x_{0:t})$ の最小化に相当

$$F(x_{0:t}) = (x_0 - \hat{x}_0)^\top \Omega_0 (x_0 - \hat{x}_0) + \sum_{(i,j) \in \mathcal{C}} e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j) \quad (19)$$

- 以上より, グラフベース SLAM が解く ポーズ調整は, 次のように書ける

$$x_{0:t}^* = \arg \min_{x_{0:t}} F(x_{0:t}) \quad (20)$$

グラフベース SLAM の定式化

- ポーズ調整 [5]

- 簡単のため、各エッジに関する項を $F_{ij}(x_i, x_j)$ と書く

$$F_{ij}(x_i, x_j) = e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j) \quad (21)$$

- また、初期姿勢 x_0 に関する項を $F_0(x_0)$ と書く

$$F_0(x_0) = (x_0 - \hat{x}_0)^\top \Omega_0 (x_0 - \hat{x}_0) \quad (22)$$

- このとき、 $F(x_{0:t})$ は次のようになる

$$F(x_{0:t}) = F_0(x_0) + \sum_{(i,j) \in \mathcal{C}} F_{ij}(x_i, x_j) \quad (23)$$

- $F_{ij}(x_i, x_j), F_0(x_0)$ はマハラノビス距離の二乗

- ポーズ調整は、**非線形最小二乗問題**

⇐ 残差 $e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$ が、 x_i, x_j について非線形

グラフベース SLAM の定式化

- ポーズ調整 [17]
 - ポーズ調整は, 関数 $F(x_{0:t})$ の最小化であった

$$x_{0:t}^* = \arg \min_{x_{0:t}} F(x_{0:t})$$

- 完全 SLAM 問題の観点からは, ポーズ調整は次のように記述した

$$x_{0:t}^* = \arg \max_{x_{0:t}} p(x_{0:t} \mid \hat{x}_0, u_{0:t}, z_{0:t})$$

- 両者は同じことを表しているはず
- ここでは, 両者の関係については考えない

③ グラフベース SLAM の定式化

- 完全 SLAM 問題
- ポーズグラフのエッジ
- ポーズ調整
- まとめ

グラフベース SLAM の定式化

- ポーズ調整のまとめ
 - 目的関数 $F(x_{0:t})$ の最小化

$$x_{0:t}^* = \arg \min_{x_{0:t}} F(x_{0:t})$$

- $x_{0:t}$ は, 全時刻におけるロボットの姿勢 (軌跡)
- $F(x_{0:t})$ は, ポーズグラフから構成できる
- 各ノードが持つ情報: 姿勢 x_t
- 各エッジが持つ情報: 観測による相対姿勢 z_{ij} と情報行列 Ω_{ij}
- 各エッジについて, マハラノビス距離の二乗 $F_{ij}(x_i, x_j)$ を最小化

$$F_{ij}(x_i, x_j) = e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j)$$

グラフベース SLAM の定式化

- ポーズ調整のまとめ

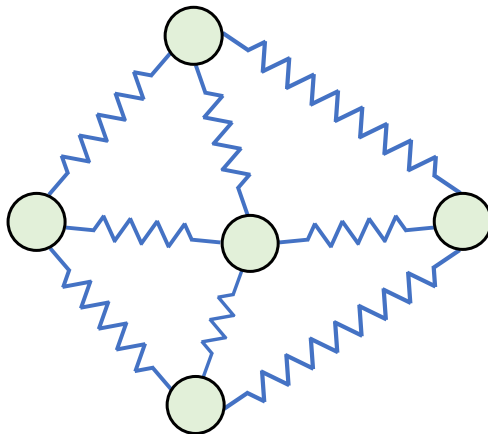
- 各エッジが持つ情報: 観測による相対姿勢 z_{ij} と情報行列 Ω_{ij}
- 各エッジについて, マハラノビス距離の二乗 $F_{ij}(x_i, x_j)$ を最小化

$$F_{ij}(x_i, x_j) = e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j)$$

- $e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$ は残差とよばれる
- z_{ij} は, 観測によって得られる, x_i と x_j の相対姿勢 (定数)
 - ⇐ スキャンマッチングや, ループ検出によって得られる
- \hat{z}_{ij} は, ポーズ調整によって修正される, x_i と x_j の相対姿勢
 - ⇐ 変数 x_i, x_j の非線形な関数
 - ⇐ 変数 x_i, x_j はノードが表現する

グラフベース SLAM の定式化

- ポーズ調整のまとめ
 - ノードの姿勢は自在に変わるが、エッジのもつ情報 (拘束) は固定される
⇐ バネで繋がれたおもりを, エネルギーが最小になるように動かす

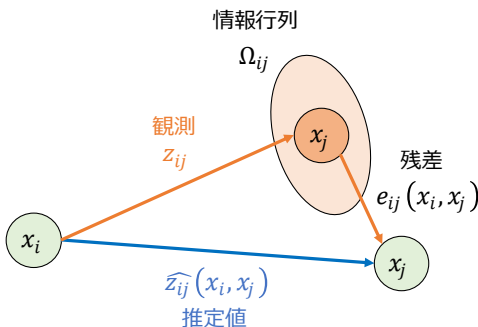


目次

- ① イントロダクション
- ② グラフベース SLAM とは
- ③ グラフベース SLAM の定式化
- ④ ポーズグラフの構築**
- ⑤ ポーズ調整の解法
- ⑥ ポーズ調整のまとめ
- ⑦ 細かな話題
- ⑧ まとめ

ポーズグラフの構築

- 相対姿勢の計算 [5, 16, 17]
 - ポーズグラフのエッジを, どのように作成するか?
 - 各エッジは, 観測に基づく相対姿勢 z_{ij} と情報行列 Ω_{ij} を保持
 - 残差 $e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$
 - 相対姿勢 z_{ij} は, 観測 (計測やループ検出) によって得る



ポーズグラフの構築

- 共分散の計算

- 各エッジは, 観測に基づく相対姿勢 z_{ij} と情報行列 Ω_{ij} を保持

- 残差の共分散 $\Sigma_{ij} = \Omega_{ij}^{-1}$ も必要

⇒ 共分散をどのようにして求めるか?

⇒ あまり詳しく知らないので, ここでは説明を省略

- 計算例

- 具体的な計算例 (解析的) [17]
- 最適化ソルバで推定する [6]
- スキャンマッチング (ICP) の誤差関数のヘッセ行列 [16, 2, 4]
- スキャンマッチングによって得た姿勢の周囲でしらみつぶし [10]
- 各スキャン点をガウス分布でモデル化 (NDT) [3] [9]

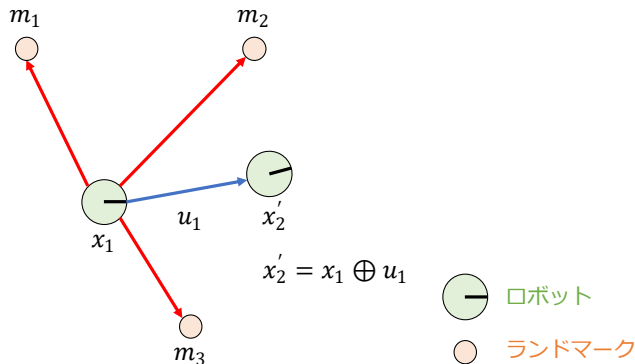
- 観測によって得た相対姿勢 z_{ij} が信頼できるとき, 共分散 Σ_{ij} は小さい

- ポーズグラフのエッジを作成する, 2つの方法を説明

- 4 ポーズグラフの構築
 - 逐次的な SLAM による構築
 - ループ検出による構築
 - まとめ

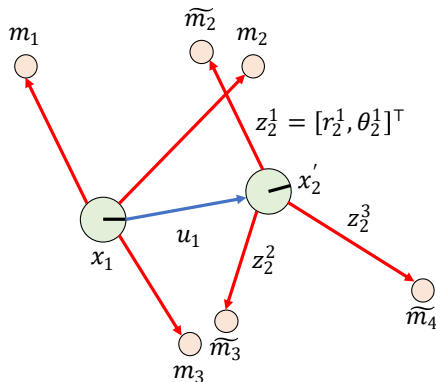
ポーズグラフの構築

- 逐次的な SLAM によるエッジの作成 (制御の反映) [16]
 - 以下の図で, ロボットが x_1 から x_2 に移動する場合を考える
 - 制御 u_1 により, ロボットが x_1 から x'_2 に移動
 - 地図 m には, 3 つのランドマーク m_1, m_2, m_3 が登録 (x_1 で観測)



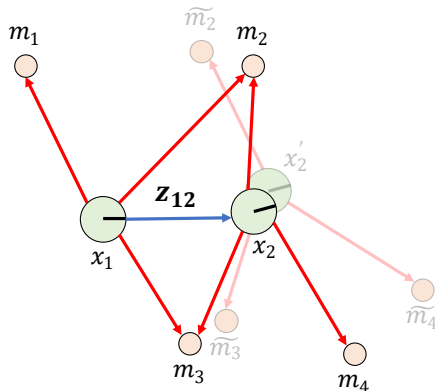
ポーズグラフの構築

- 逐次的な SLAM によるエッジの作成 (スキャンマッチング) [16]
 - 時刻 $t = 2$ で, 3つのランドマーク $\tilde{m}_2, \tilde{m}_3, \tilde{m}_4$ を観測 (計測 z_t)
 - ランドマーク 2, 3 は本来は同一の位置だが, 重なっていない
⇐ 制御 u_t と計測 z_t に含まれる誤差のため ($m_2 \neq \tilde{m}_2, m_3 \neq \tilde{m}_3$)



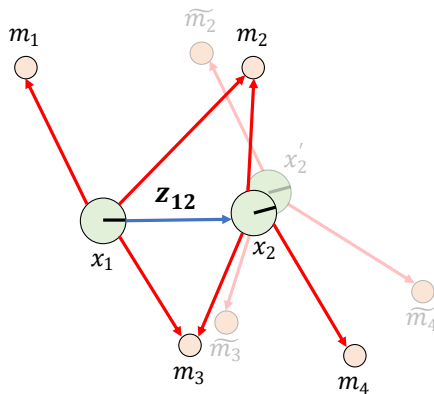
ポーズグラフの構築

- 逐次的な SLAM によるエッジの作成 (スキャンマッチング) [16]
 - ランドマークと地図の重ね合わせにより, 姿勢を改良 ($x'_2 \rightarrow x_2$)
 - 2つのランドマーク \tilde{m}_2, \tilde{m}_3 が, 地図に含まれる m_2, m_3 と重なる
 - 時刻 $t = 2$ における, 最終的な姿勢 x_2 が得られる



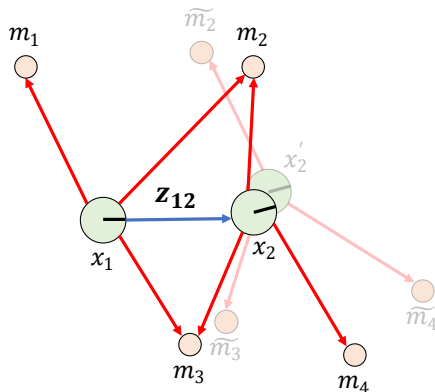
ポーズグラフの構築

- 逐次的な SLAM によるエッジの作成 (ノード追加) [16]
 - 時刻 $t = 2$ における, 最終的な姿勢 x_2 が得られる
 - 姿勢 x_2 を, ポーズグラフにノード x_2 として追加



ポーズグラフの構築

- 逐次的な SLAM によるエッジの作成 (エッジ追加) [16]
 - x_1 と x_2 との**相対姿勢** $z_{12} = x_2 \ominus x_1$ が計算できる
 - 何らかの方法で, 共分散 Σ_{12} を得る (x_2 の不確実性を反映)
 - 相対姿勢 z_{12} と情報行列 $\Omega_{12} = \Sigma_{12}^{-1}$ を, ポーズグラフの**エッジ**に追加

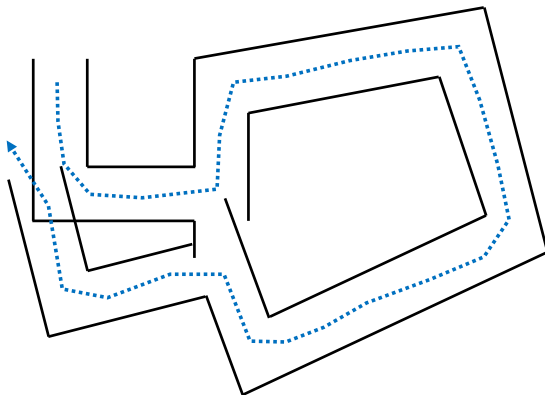


4 ポーズグラフの構築

- 逐次的な SLAM による構築
- ループ検出による構築
- まとめ

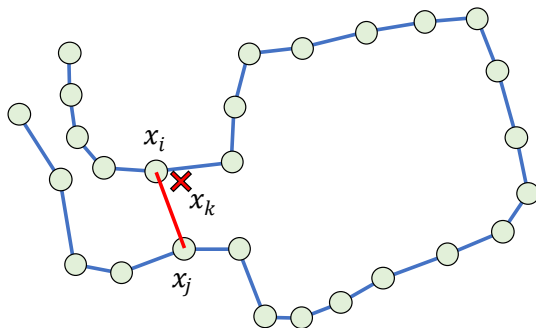
ポーズグラフの構築

- ループ検出によるエッジの作成 [16]
 - 逐次的な SLAM だけでは、誤差が累積する
 - 以前訪れた場所に戻ってきたのに、地図上で重ならない
 - 地図上に、同じ場所が複数出現する



ポーズグラフの構築

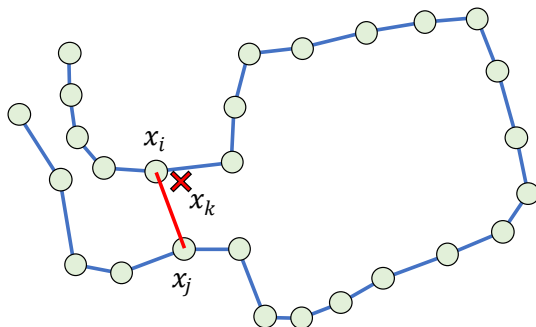
- ループ検出によるエッジの作成 (ループ検出) [16]
 - 何らかの方法で、ループを検出したとする
 - ノード x_j と対応する点を, ノード x_i を始点として探索
 $\Rightarrow x_j$ が, 地図上の x_k と対応することが判明



— オドメトリの拘束 — ループの拘束

ポーズグラフの構築

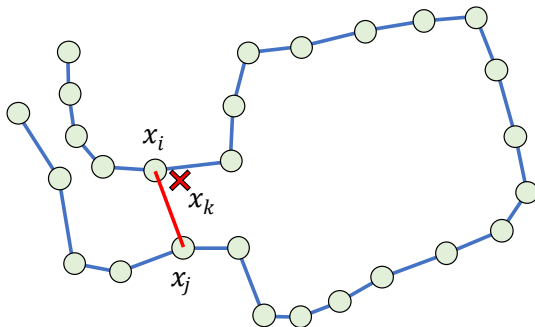
- ループ検出によるエッジの作成 (ループ検出) [16]
 - x_j が, 地図上の x_k と対応することが判明
 - ノード x_i と x_j の間に, **ループ**を表すエッジを追加



— オドメトリの拘束 — ループの拘束

ポーズグラフの構築

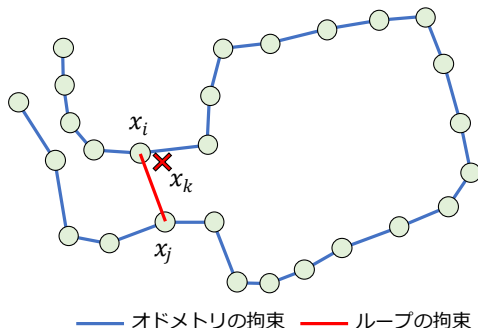
- ループ検出によるエッジの作成 (エッジ追加) [16]
 - ノード x_i と x_j の間に, **ループ**を表すエッジを追加
 - エッジの相対姿勢 z_{ij} は, x_i と x_k の**相対姿勢**とする
⇐ x_i と x_j の相対姿勢ではないことに注意



— オドメトリの拘束 — ループの拘束

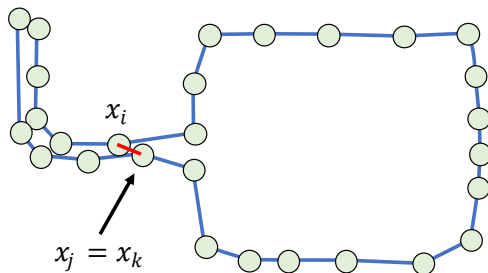
ポーズグラフの構築

- ループ検出によるエッジの作成 (エッジ追加) [16]
 - ノード x_i と x_j の間に, **ループ**を表すエッジを追加
 - 逐次処理により, ノードの姿勢 x_i と x_j は既に求まっている
 - 逐次処理により求めた, **ノード** x_i と x_j の相対姿勢
⇐ x_i と x_k の相対姿勢 (実際の**エッジの観測**) とは**大きく異なる**



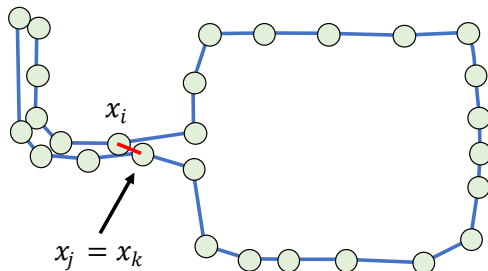
ポーズグラフの構築

- ループ検出によるエッジの作成 (ポーズ調整) [16]
 - ループを表すエッジを追加した後, **ポーズ調整**を実行
 - エッジの拘束を満たすように, ポーズグラフを最適化
 - ノード x_j が x_k に重なるように移動 (ノードとエッジの矛盾を解消)
⇒ それに合わせて, 他のノードも修正される



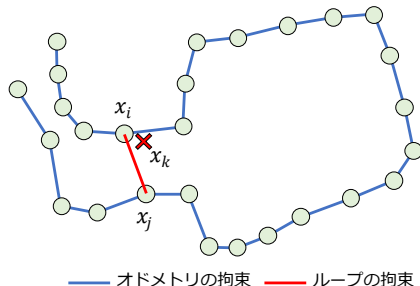
ポーズグラフの構築

- ループ検出によるエッジの作成 (ポーズ調整) [16]
 - ループを表すエッジを追加した後, **ポーズ調整**を実行
 - 逐次処理により求めた姿勢を初期値として, 反復実行される (後述)
 - ポーズ調整: 関数 $F(x_{0:t})$ の, ノードの姿勢 $x_{0:t}$ に関する最小化



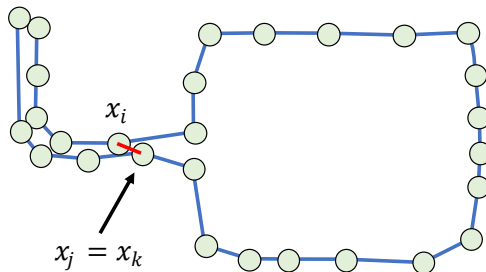
ポーズグラフの構築

- ループ検出によるエッジの作成 (補足; ポーズ調整前) [16]
 - エッジの残差: $e_{ij} = z_{ij} - \hat{z}_{ij}(x_i, x_j)$ (ノードとエッジの乖離具合)
 - x_i, x_j は逐次処理により求めたノードの姿勢 (初期値)
 - $\Rightarrow \hat{z}_{ij}(x_i, x_j) = x_j \ominus x_i$ は, 2つのノード x_i, x_j の相対姿勢
 - \Rightarrow エッジがもつ, 観測により得た z_{ij} は, x_i, x_k の相対姿勢
 - $\Rightarrow z_{ij}$ と \hat{z}_{ij} は大きく異なるので, 残差 e_{ij} は大きい



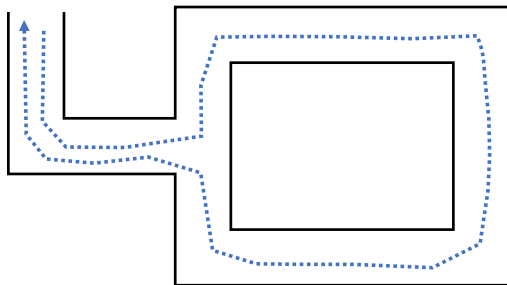
ポーズグラフの構築

- ループ検出によるエッジの作成 (補足; ポーズ調整~~後~~) [16]
 - エッジの残差: $e_{ij} = z_{ij} - \hat{z}_{ij}(x_i, x_j)$
 - ポーズ調整後は, ノード x_j が x_k に近づく
 - \Rightarrow ノード x_i, x_j の相対姿勢が, 観測 $z_{ij}(x_i, x_k$ の相対姿勢) に近づく
 - $\Rightarrow \hat{z}_{ij}$ が z_{ij} に近づくので, 残差 e_{ij} は小さくなる



ポーズグラフの構築

- ループ検出によるエッジの作成 (ポーズ調整) [16]
 - ポーズ調整により, ロボットの軌跡 (全ノードの姿勢) が修正される
 - 新しい軌跡を使って, 地図を作成し直す
 - 大域的に整合性の取れた, 精度の良い地図が得られる



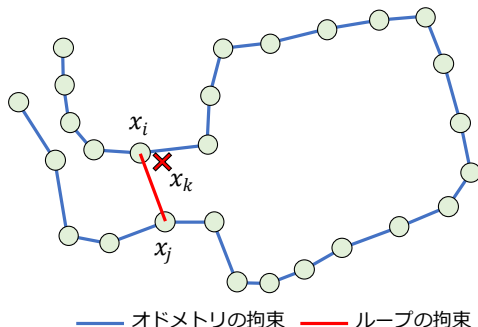
4 ポーズグラフの構築

- 逐次的な SLAM による構築
- ループ検出による構築
- まとめ

ポーズグラフの構築

- まとめ

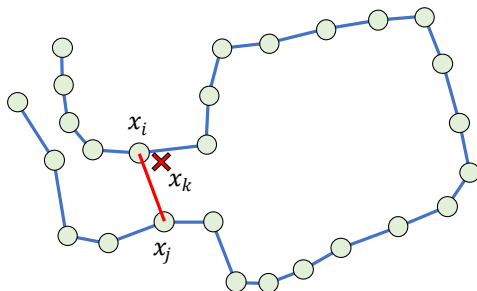
- ポーズグラフのエッジには 2 種類ある
 - ⇒ オドメトリを表すエッジと、ループを表すエッジ
- オドメトリを表すエッジは、隣接する 2 時刻のノードを結ぶ
 - ⇔ ループを表すエッジは、時間的に離れた 2 つのノードを結ぶ



ポーズグラフの構築

- まとめ

- オドメトリを表すエッジは, 逐次処理により作成
⇔ ループを表すエッジは, ループ検出により作成
- ループを表すエッジのおかげで, ポーズ調整が可能になる
⇔ オドメトリを表すエッジだけでは, ポーズ調整はできない



— オドメトリの拘束 — ループの拘束

目次

- ① イントロダクション
- ② グラフベース SLAM とは
- ③ グラフベース SLAM の定式化
- ④ ポーズグラフの構築
- ⑤ ポーズ調整の解法**
- ⑥ ポーズ調整のまとめ
- ⑦ 細かな話題
- ⑧ まとめ

- 5 ポーズ調整の解法
 - ポーズ調整の概要
 - ポーズ調整の入出力
 - ポーズ調整のアルゴリズムの導出

ポーズ調整の解法

- ポーズグラフを基に最適化問題を構成 [5, 7, 16, 17]
 - ポーズグラフが解くのは, **非線形最小二乗問題**
 - ポーズグラフのもつ情報を, 多数の**連立方程式**に変換する
 - ポーズ調整は, **反復的**なアルゴリズム
 - ⇐ ガウス・ニュートン法 (Gauss-Newton)
 - ⇐ レーベンバーグ・マーカート法 (Levenberg-Marquardt)
 - 連立方程式は大規模かつ**スパース** (2次元ならノード数の3倍)
 - ⇐ 疎行列コレスキー分解 (Sparse Cholesky Factorization)
 - ⇐ 前処理付き共役勾配法 (Preconditioned Conjugate Gradient)
 - スパース性は, ポーズグラフが疎であることに由来
 - ⇐ エッジ数は, ノード数より少し多い程度

- 5 ポーズ調整の解法
 - ポーズ調整の概要
 - ポーズ調整の入出力
 - ポーズ調整のアルゴリズムの導出

ポーズ調整の解法

- ポーズ調整の入出力

- 入力:** ポーズグラフ

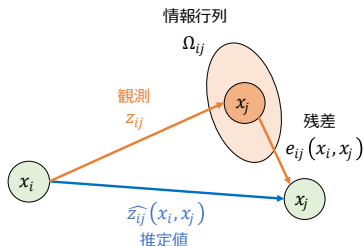
- ⇐ ノードは姿勢 x_t を表現

- ⇐ エッジは観測に基づく相対姿勢 z_{ij} と、情報行列 Ω_{ij} を保持

- ノードの姿勢 x_t は変数**

- ⇔ エッジの相対姿勢 z_{ij} と情報行列 Ω_{ij} は**定数**

- 出力:** 修正されたノードの姿勢 $x_{0:t}$



- 5 ポーズ調整の解法
 - ポーズ調整の概要
 - ポーズ調整の入出力
 - ポーズ調整のアルゴリズムの導出

ポーズ調整の解法

- ポーズ調整とは [5, 17]
 - 関数 $F(x_{0:t})$ を, ノードの姿勢 $x_{0:t}$ について最小化する問題

$$x_{0:t}^* = \arg \min_{x_{0:t}} F(x_{0:t})$$

- 関数 $F(x_{0:t})$ は, 次のように表現される

$$F(x_{0:t}) = F_0(x_0) + \sum_{(i,j) \in \mathcal{C}} F_{ij}(x_i, x_j)$$

$$F_0(x_0) = (x_0 - \hat{x}_0)^\top \Omega_0 (x_0 - \hat{x}_0)$$

$$F_{ij}(x_i, x_j) = e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j)$$

- $F_0(x_0)$ は, 初期姿勢 x_0 を定数 \hat{x}_0 に固定する役割
- 情報行列 Ω_0 は対角行列で, 各成分は非常に大きな値に設定 (定数)

ポーズ調整の解法

- ポーズ調整とは [5, 17]
 - 関数 $F(x_{0:t})$ は, 次のように表現される

$$F(x_{0:t}) = F_0(x_0) + \sum_{(i,j) \in \mathcal{C}} F_{ij}(x_i, x_j)$$

$$F_0(x_0) = (x_0 - \hat{x}_0)^\top \Omega_0 (x_0 - \hat{x}_0)$$

$$F_{ij}(x_i, x_j) = e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j)$$

- $F_{ij}(x_i, x_j)$ は, ノード x_i, x_j を結ぶエッジ (i, j) に対応する項
- ポーズグラフ内の全エッジに対して, F_{ij} を足し合わせる

ポーズ調整の解法

- ポーズ調整とは [5, 17]
 - 関数 $F(x_{0:t})$ は, 次のように表現される

$$F(x_{0:t}) = F_0(x_0) + \sum_{(i,j) \in \mathcal{C}} F_{ij}(x_i, x_j)$$

$$F_{ij}(x_i, x_j) = e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j)$$

- $e_{ij}(x_i, x_j)$ は, 各エッジの**残差**

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$$

- z_{ij} は各エッジがもつ**相対姿勢 (定数)**
- $\hat{z}_{ij}(x_i, x_j)$ は, エッジの両端のノード x_i, x_j の**相対姿勢**
⇐ ノードの姿勢 x_i, x_j は**変数**であり, 最適化の対象

ポーズ調整の解法

- ポーズ調整とは [5, 17]
 - $e_{ij}(x_i, x_j)$ は, 各エッジの**残差**

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$$

- z_{ij} は各エッジがもつ**相対姿勢 (定数)**
- $\hat{z}_{ij}(x_i, x_j)$ は, エッジの両端のノード x_i, x_j の**相対姿勢**
- $x_i = [\xi_i^x, \xi_i^y, \xi_i^\theta]^\top$, $x_j = [\xi_j^x, \xi_j^y, \xi_j^\theta]^\top$ とすると,

$$\hat{z}_{ij}(x_i, x_j) = x_j \ominus x_i = \begin{bmatrix} \cos \xi_i^\theta & \sin \xi_i^\theta & 0 \\ -\sin \xi_i^\theta & \cos \xi_i^\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \xi_j^x - \xi_i^x \\ \xi_j^y - \xi_i^y \\ \xi_j^\theta - \xi_i^\theta \end{bmatrix}.$$

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (初期値) [5]
 - ノードの姿勢 $x_{0:t}$ は, SLAM の逐次実行によって既に得られている
⇒ これを**初期値**として採用 ($\check{x}_{0:t}$ とする)
 - 各姿勢 x_i を, **初期値** \check{x}_i と, **初期値からの差分** Δx_i を用いて表現

$$x_i = \check{x}_i + \Delta x_i \quad (24)$$

- 続いて, 姿勢 $x_{0:t}$ についての関数 $F(x_{0:t})$ を書き換える
⇒ **差分** $\Delta x_0, \dots, \Delta x_t$ についての関数に変換
⇒ $F(x_{0:t})$ を **Δx に関して**最小化

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (テイラー展開) [5]
 - 初期値 \check{x}_i, \check{x}_j のまわりで、残差 $e_{ij}(x_i, x_j)$ をテイラー展開 (1 次)

$$e_{ij}(x_i, x_j) = e_{ij}(\check{x}_i + \Delta x_i, \check{x}_j + \Delta x_j) \quad (25)$$

$$\simeq e_{ij}(\check{x}_i, \check{x}_j) + J_i \Delta x_i + J_j \Delta x_j \quad (26)$$

- 微小な変化であれば ($\Delta x_i, \Delta x_j$ が小さければ) 成り立つ
 - ここで、 J_i, J_j は 3×3 のヤコビ行列
- $\Rightarrow x_i = \check{x}_i, x_j = \check{x}_j$ の下での、 $e_{ij}(x_i, x_j)$ の x_i, x_j に関する偏微分

$$J_i = \left. \frac{\partial e_{ij}(x_i, x_j)}{\partial x_i} \right|_{x_i=\check{x}_i, x_j=\check{x}_j} \quad (27)$$

$$J_j = \left. \frac{\partial e_{ij}(x_i, x_j)}{\partial x_j} \right|_{x_i=\check{x}_i, x_j=\check{x}_j} \quad (28)$$

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (表記の簡略化) [5]
 - 姿勢 $x_{0:t}$ を全て繋げて, 1 本の細長いベクトル x にする

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_t \end{bmatrix} \quad (29)$$

- ポーズグラフのノード数を T とする ($T = t + 1$)
 $\Rightarrow x$ は **3T 次元**ベクトル (x_0, \dots, x_t は 3 次元)
- $e_{ij}(x_i, x_j)$ は x の関数とみなせる (実際には x_i, x_j のみの関数)

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (残差の式変形) [5]
 - $e_{ij}(x_i, x_j)$ は x の関数でもあるので, 次のように書ける

$$e_{ij}(x_i, x_j) = e_{ij}(\check{x}_i + \Delta x_i, \check{x}_j + \Delta x_j) \quad (30)$$

$$= e_{ij}(\check{x} + \Delta x) \quad (31)$$

$$\simeq e_{ij}(\check{x}) + J_{ij} \Delta x \quad (32)$$

- 但し $\check{x}, \Delta x$ は, $\check{x}_0, \dots, \check{x}_t, \Delta x_0, \dots, \Delta x_t$ を全て繋げたもの
 $\Rightarrow \check{x}, \Delta x$ も **3T 次元**ベクトル
 $\Rightarrow x = \check{x} + \Delta x$ であることに注意

$$\check{x} = [\check{x}_0^\top \check{x}_1^\top \cdots \check{x}_t^\top]^\top \quad (33)$$

$$\Delta x = [\Delta x_0^\top \Delta x_1^\top \cdots \Delta x_t^\top]^\top \quad (34)$$

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (残差の式変形) [5]
 - $e_{ij}(x_i, x_j)$ は次のように書ける

$$e_{ij}(x_i, x_j) \simeq e_{ij}(\check{x}) + J_{ij} \Delta x$$

- J_{ij} は $3 \times 3T$ の横長なヤコビ行列

$$J_{ij} = [0 \ \cdots \ 0 \ J_i \ 0 \ \cdots \ 0 \ J_j \ 0 \ \cdots \ 0] \quad (35)$$

- J_{ij} は, ノード i と j に対応する部分に, ヤコビ行列が入る
 - $\Leftarrow 3i$ から $3i + 2$ 列までが J_i , そして, $3j$ から $3j + 2$ 列までが J_j
 - \Leftarrow それ以外の要素は, 全て 0
- 以下は簡単に確認できる

$$J_{ij} \Delta x = J_i \Delta x_i + J_j \Delta x_j \quad (36)$$

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (項 F_{ij} の式変形) [5]
 - $F_{ij}(x_i, x_j)$ は以下のものであった

$$F_{ij}(x_i, x_j) = e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j)$$

- これも x の関数として、次のように書き直せる

$$F_{ij}(x_i, x_j) = F_{ij}(\check{x}_i + \Delta x_i, \check{x}_j + \Delta x_j) \quad (37)$$

$$= F_{ij}(\check{x} + \Delta x) \quad (38)$$

$$= e_{ij}(\check{x} + \Delta x)^\top \Omega_{ij} e_{ij}(\check{x} + \Delta x) \quad (39)$$

- 先程のテイラー展開を代入する

$$F_{ij}(\check{x} + \Delta x) \simeq (e_{ij}(\check{x}) + J_{ij} \Delta x)^\top \Omega_{ij} (e_{ij}(\check{x}) + J_{ij} \Delta x)$$

- ポーズ調整のアルゴリズムの導出 (項 F_{ij} の式変形) [5]
 - $F_{ij}(\check{x} + \Delta x)$ を, Δx の関数として表す

$$\begin{aligned} & F_{ij}(\check{x} + \Delta x) \\ \simeq & (e_{ij}(\check{x}) + J_{ij}\Delta x)^\top \Omega_{ij} (e_{ij}(\check{x}) + J_{ij}\Delta x) \\ = & e_{ij}(\check{x})^\top \Omega_{ij} e_{ij}(\check{x}) + 2e_{ij}(\check{x})^\top \Omega_{ij} J_{ij} \Delta x + \Delta x^\top J_{ij}^\top \Omega_{ij} J_{ij} \Delta x \\ = & c_{ij} + 2b_{ij}^\top \Delta x + \Delta x^\top H_{ij} \Delta x \end{aligned} \quad (40)$$

- 但し, 係数 c_{ij}, b_{ij}, H_{ij} は次のように定める ($\Omega_{ij}^\top = \Omega_{ij}$ に注意)

$$c_{ij} = e_{ij}(\check{x})^\top \Omega_{ij} e_{ij}(\check{x}) \quad (41)$$

$$b_{ij} = J_{ij}^\top \Omega_{ij} e_{ij}(\check{x}) \quad (42)$$

$$H_{ij} = J_{ij}^\top \Omega_{ij} J_{ij} \quad (43)$$

- c_{ij} はスカラー / b_{ij} は $3T$ 次元ベクトル / H_{ij} は $3T$ 次正方行列

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (係数の表現) [5]
 - ここで, 係数 b_{ij} , H_{ij} の中身を調べてみる

$$F_{ij}(\check{x} + \Delta x) \simeq c_{ij} + 2b_{ij}^\top \Delta x + \Delta x^\top H_{ij} \Delta x$$

- $3 \times 3T$ ヤコビ行列 J_{ij} は次のようであった (J_i, J_j 以外は全て 0)

$$J_{ij} = [\cdots J_i \cdots J_j \cdots]$$

- J_i, J_j は, ノード x_i, x_j に関するヤコビ行列 (3×3)
⇐ 初期値 $x = \check{x}$ のもとでの, 残差 $e_{ij}(x)$ の偏微分

$$\begin{aligned} J_i &= \left. \frac{\partial e_{ij}(x_i, x_j)}{\partial x_i} \right|_{x_i=\check{x}_i, x_j=\check{x}_j} = \left. \frac{\partial e_{ij}(x)}{\partial x_i} \right|_{x=\check{x}} \\ J_j &= \left. \frac{\partial e_{ij}(x_i, x_j)}{\partial x_j} \right|_{x_i=\check{x}_i, x_j=\check{x}_j} = \left. \frac{\partial e_{ij}(x)}{\partial x_j} \right|_{x=\check{x}} \end{aligned}$$

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (係数 b_{ij} , H_{ij} の表現) [5]
 - 係数 b_{ij} の中身を調べてみる

$$\begin{aligned} b_{ij} &= J_{ij}^\top \Omega_{ij} e_{ij}(\check{x}) \\ &= [\cdots J_i \cdots J_j \cdots]^\top \Omega_{ij} e_{ij}(\check{x}) \\ &= \begin{bmatrix} \vdots \\ J_i^\top \\ \vdots \\ J_j^\top \\ \vdots \end{bmatrix} \Omega_{ij} e_{ij}(\check{x}) = \begin{bmatrix} \vdots \\ J_i^\top \Omega_{ij} e_{ij}(\check{x}) \\ \vdots \\ J_j^\top \Omega_{ij} e_{ij}(\check{x}) \\ \vdots \end{bmatrix} \end{aligned} \quad (44)$$

- ノード i に対応するブロック: $3i$ から $3i + 2$ 行目まで
- ノード j に対応するブロック: $3j$ から $3j + 2$ 行目まで

ポーズ調整の解法

- 続いて, 係数 H_{ij} の中身を調べてみる

$$\begin{aligned} H_{ij} &= J_{ij}^\top \Omega_{ij} J_{ij} \\ &= \begin{bmatrix} \vdots \\ J_i^\top \\ \vdots \\ J_j^\top \\ \vdots \end{bmatrix} \Omega_{ij} [\cdots J_i \cdots J_j \cdots] \\ &= \begin{bmatrix} \ddots & & & & \\ & J_i^\top \Omega_{ij} J_i & \cdots & J_i^\top \Omega_{ij} J_j & \\ & \vdots & \ddots & \vdots & \\ & J_j^\top \Omega_{ij} J_i & \cdots & J_j^\top \Omega_{ij} J_j & \\ & & & & \ddots \end{bmatrix} \end{aligned} \quad (45)$$

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (項 F_0 の式変形) [5]

- $F_{ij}(\check{x} + \Delta x)$ は, Δx の関数として書いた

$$F_{ij}(\check{x} + \Delta x) \simeq c_{ij} + 2b_{ij}^\top \Delta x + \Delta x^\top H_{ij} \Delta x$$

- ポーズ調整で最適化したい関数 $F(x_{0:t})$ は, 次のようであった

$$F(x_{0:t}) = F_0(x_0) + \sum_{(i,j) \in \mathcal{C}} F_{ij}(x_i, x_j)$$

- $F_0(x_0)$ は, 初期姿勢 x_0 を \hat{x}_0 に固定するための項
- 項 $F_0(x_0)$ も F_{ij} と同じく, Δx の関数として書いておく

$$F_0(x_0) = (x_0 - \hat{x}_0)^\top \Omega_0 (x_0 - \hat{x}_0)$$

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (項 F_0 の式変形) [5, 17]
 - 最初に, $F_0(x_0)$ を, Δx_0 の関数として表す

$$F_0(x_0) = F_0(\check{x}_0 + \Delta x_0) \quad (46)$$

$$\begin{aligned} &= (\check{x}_0 + \Delta x_0 - \hat{x}_0)^\top \Omega_0 (\check{x}_0 + \Delta x_0 - \hat{x}_0) \\ &= \Delta x_0^\top \Omega_0 \Delta x_0 + 2(\check{x}_0 - \hat{x}_0)^\top \Omega_0 \Delta x_0 + \\ &\quad (\check{x}_0 - \hat{x}_0)^\top \Omega_0 (\check{x}_0 - \hat{x}_0) \\ &= \Delta x_0^\top \Omega_0 \Delta x_0 \end{aligned} \quad (47)$$

- \check{x}_0 は x_0 の初期値
 - \Rightarrow アルゴリズムの実行前は $x_0 = \hat{x}_0$ であるので, $\check{x}_0 = \hat{x}_0$
 - $\Rightarrow \check{x}_0 - \hat{x}_0 = 0$ が成り立つ

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (項 F_0 の式変形) [5]
 - 最後に, $F_0(x_0)$ を, Δx の関数として表す

$$F_0(x) = F_0(\check{x} + \Delta x) = \Delta x^\top \tilde{\Omega}_0 \Delta x \quad (48)$$

- 上記の $\tilde{\Omega}_0$ は, $3T \times 3T$ 行列
 $\Rightarrow 3 \times 3$ 行列 Ω_0 を左上に置き, 残りの部分を 0 で埋めて拡張

$$\tilde{\Omega}_0 = \begin{bmatrix} \Omega_0 & & \\ & \ddots & \\ & & 0 \end{bmatrix} \quad (49)$$

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (関数 $F(x_{0:t})$ の式変形) [5]
 - $F_{ij}(x_i, x_j)$ を, Δx の関数 $F_{ij}(\check{x} + \Delta x)$ として表現した

$$F_{ij}(\check{x} + \Delta x) \simeq c_{ij} + 2b_{ij}^\top \Delta x + \Delta x^\top H_{ij} \Delta x$$

- 同様に $F_0(x_0)$ も, Δx の関数 $F_0(\check{x} + \Delta x)$ として表現した

$$F_0(\check{x} + \Delta x) = \Delta x^\top \tilde{\Omega}_0 \Delta x$$

- 最後に, 関数 $F(x_{0:t})$ を, 差分 Δx の関数に書き換える (当初の目標)

$$F(x_{0:t}) = F_0(x_0) + \sum_{(i,j) \in \mathcal{C}} F_{ij}(x_i, x_j)$$

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (関数 $F(x_{0:t})$ の式変形) [5]
 - 関数 $F(x_{0:t})$ を, Δx の関数として表す

$$F(x) = F(\check{x} + \Delta x) \quad (50)$$

$$= F_0(\check{x} + \Delta x) + \sum_{(i,j) \in \mathcal{C}} F_{ij}(\check{x} + \Delta x) \quad (51)$$

$$\simeq \Delta x^\top \tilde{\Omega}_0 \Delta x + \sum_{(i,j) \in \mathcal{C}} (c_{ij} + 2b_{ij}^\top \Delta x + \Delta x^\top H_{ij} \Delta x) \quad (52)$$

$$= c + 2b^\top \Delta x + \Delta x^\top H \Delta x \quad (53)$$

- 但し, 係数 c, b, H は次のように定める

$$c = \sum_{(i,j) \in \mathcal{C}} c_{ij}, \quad b = \sum_{(i,j) \in \mathcal{C}} b_{ij}, \quad H = \tilde{\Omega}_0 + \sum_{(i,j) \in \mathcal{C}} H_{ij} \quad (54)$$

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (更新式の導出) [5]
 - 関数 $F(x_{0:t}) = F(\check{x} + \Delta x)$ を最小化する Δx を求める

$$\frac{\partial F(\check{x} + \Delta x)}{\partial \Delta x} = 0 \quad (55)$$

$$\Rightarrow \frac{\partial}{\partial \Delta x} (c + 2b^\top \Delta x + \Delta x^\top H \Delta x) = 0 \quad (56)$$

$$\Rightarrow 2b + 2H\Delta x = 0 \quad (57)$$

- 従って、最適な Δx に関して、次の線形方程式が得られる

$$H\Delta x^* = -b \quad (58)$$

- 解 Δx^* を初期値 \check{x} に足しあわせて、全ノードの姿勢 x^* を得る

$$x^* = \check{x} + \Delta x^* \quad (59)$$

ポーズ調整の解法

- ポーズ調整のアルゴリズムの導出 (更新式の導出) [5]
 - 解 Δx^* を初期値 \check{x} に足しあわせて, 全ノードの姿勢 x^* を得る

$$x^* = \check{x} + \Delta x^*$$

- $F_{ij}(\check{x} + \Delta x)$ は, テイラー展開により近似
 - ⇐ F_{ij} は, 初期値 \check{x} のまわりで線形化
 - ⇐ F_{ij} の近似は, 初期値 \check{x} の近傍でのみ正確
- 求めた解 x^* を初期値 \check{x} として, 反復計算を行う
- 長かった!

目次

- ① イントロダクション
- ② グラフベース SLAM とは
- ③ グラフベース SLAM の定式化
- ④ ポーズグラフの構築
- ⑤ ポーズ調整の解法
- ⑥ ポーズ調整のまとめ
- ⑦ 細かな話題
- ⑧ まとめ

- 6 ポーズ調整のまとめ
 - ポーズ調整の入出力
 - ポーズ調整のアルゴリズム
 - 1/4: 方程式の係数の計算
 - 2/4: 方程式を解く
 - 3/4: 解の更新
 - 4/4: 収束判定
 - ポーズ調整のためのライブラリ

ポーズ調整のまとめ

- ポーズ調整の**入力**

- ポーズグラフの, 全ノードの姿勢 $x_{0:t}$
⇐ SLAM の逐次処理により得られ, **初期値** \check{x} として使用
- ポーズグラフの, 全エッジのもつ観測 z_{ij} と, 情報行列 Ω_{ij}
⇐ 観測 z_{ij} は, ノード i と j の相対姿勢 (**拘束**)

- ポーズ調整の**出力**

- 改良された, 全ノードの姿勢 $x_{0:t}^*$

6 ポーズ調整のまとめ

- ポーズ調整の入出力
- ポーズ調整のアルゴリズム
- 1/4: 方程式の係数の計算
- 2/4: 方程式を解く
- 3/4: 解の更新
- 4/4: 収束判定
- ポーズ調整のためのライブラリ

ポーズ調整のまとめ

- ポーズ調整のアルゴリズム (概略) [5]

- 1 目的関数 $F(\check{x} + \Delta x)$ の係数 H, b を計算する

$$F(\check{x} + \Delta x) \simeq c + 2b^\top \Delta x + \Delta x^\top H \Delta x$$

- 2 係数 H, b から構成した次の方程式を, Δx について解く

$$H \Delta x = -b$$

- 3 方程式の解 Δx^* を使って, 全ノードの姿勢 x を更新する

$$x^* = \check{x} + \Delta x^*$$

- 4 目的関数 $F(x)$ の値が収束したら終了

⇒ そうでなければ, 更新された姿勢 x^* を初期値 \check{x} として, (1) に戻る

- **ガウス・ニュートン法**の枠組み

6 ポーズ調整のまとめ

- ポーズ調整の入出力
- ポーズ調整のアルゴリズム
- 1/4: 方程式の係数の計算
- 2/4: 方程式を解く
- 3/4: 解の更新
- 4/4: 収束判定
- ポーズ調整のためのライブラリ

ポーズ調整のまとめ

- ポーズ調整のアルゴリズム (1/4: 係数 H, b の計算) [5]
 - $3T \times 3T$ 行列 H は, 次のように記述される

$$H = \tilde{\Omega}_0 + \sum_{(i,j) \in \mathcal{C}} H_{ij}, \quad \tilde{\Omega}_0 = \begin{bmatrix} \Omega_0 & & \\ & \ddots & \\ & & \end{bmatrix}$$
$$H_{ij} = \begin{bmatrix} \ddots & & & & \\ & \underbrace{\begin{matrix} [3i, 3i+2] \text{ 行} \\ [3i, 3i+2] \text{ 列} \end{matrix}}_{J_i^\top \Omega_{ij} J_i} & \cdots & \underbrace{\begin{matrix} [3i, 3i+2] \text{ 行} \\ [3j, 3j+2] \text{ 列} \end{matrix}}_{J_i^\top \Omega_{ij} J_j} & \\ & \vdots & \ddots & \vdots & \\ & \underbrace{\begin{matrix} [3j, 3j+2] \text{ 行} \\ [3i, 3i+2] \text{ 列} \end{matrix}}_{J_j^\top \Omega_{ij} J_i} & \cdots & \underbrace{\begin{matrix} [3j, 3j+2] \text{ 行} \\ [3j, 3j+2] \text{ 列} \end{matrix}}_{J_j^\top \Omega_{ij} J_j} & \\ & & & \ddots & \end{bmatrix}$$

ポーズ調整のまとめ

- ポーズ調整のアルゴリズム (1/4: 係数 H, b の計算) [5]
 - $3T$ 次ベクトル b は, 次のように記述される

$$b = \sum_{(i,j) \in \mathcal{C}} b_{ij}$$
$$b_{ij} = \begin{bmatrix} \vdots \\ \underbrace{J_i^\top \Omega_{ij} e_{ij}(\check{x})}_{[3i, 3i+2] \text{ 行}} \\ \vdots \\ \underbrace{J_j^\top \Omega_{ij} e_{ij}(\check{x})}_{[3j, 3j+2] \text{ 行}} \\ \vdots \end{bmatrix}$$

- 係数 H, b はいずれも **スパース**

ポーズ調整のまとめ

- ポーズ調整のアルゴリズム (1/4: 係数 H, b の計算) [5]

- 1 初期化: $b = 0, H = 0$

- 2 各ノード $(i, j) \in \mathcal{C}$ について, 以下を順に実行

- 3 3×3 のヤコビ行列 J_i, J_j を計算

- ⇐ 初期値 $x = \check{x}$ のもとでの, 残差 $e_{ij}(x)$ の偏微分

$$J_i = \left. \frac{\partial e_{ij}(x)}{\partial x_i} \right|_{x=\check{x}}, \quad J_j = \left. \frac{\partial e_{ij}(x)}{\partial x_j} \right|_{x=\check{x}}$$

- 4 残差 $e_{ij}(\check{x}) = z_{ij} - \hat{z}_{ij}(\check{x})$ を計算

- ⇐ $\hat{z}_{ij}(\check{x}) = \hat{z}_{ij}(\check{x}_i, \check{x}_j) = \check{x}_j \ominus \check{x}_i$ は, \check{x}_i と \check{x}_j の相対姿勢

5 行列 H の 4 つのブロックを更新

$$\begin{aligned} H_{[ii]} &\leftarrow H_{[ii]} + J_i^\top \Omega_{ij} J_i, & H_{[ij]} &\leftarrow H_{[ij]} + J_i^\top \Omega_{ij} J_j \\ H_{[ji]} &\leftarrow H_{[ji]} + J_j^\top \Omega_{ij} J_i, & H_{[jj]} &\leftarrow H_{[jj]} + J_j^\top \Omega_{ij} J_j \end{aligned}$$

6 ベクトル b を更新

$$b_{[i]} \leftarrow b_{[i]} + J_i^\top \Omega_{ij} e_{ij}(\check{x}), \quad b_{[j]} \leftarrow b_{[j]} + J_j^\top \Omega_{ij} e_{ij}(\check{x})$$

7 最後に、初期姿勢 x_0 を \hat{x}_0 に固定

$$H_{[00]} \leftarrow H_{[00]} + \Omega_0$$

- $H_{[kl]}$ は、行列 H の (k, l) ブロック (3×3 の部分行列)
 $\Leftarrow [3k, 3k + 2]$ 行目と, $[3l, 3l + 2]$ 列目
- $b_{[k]}$ は、ベクトル b の $[3k, 3k + 2]$ 行目

ポーズ調整のまとめ

- ポーズ調整のアルゴリズム (1/4: 係数 H, b の計算) [5]
 - 各ノードについて計算される H_{ij} はスパース
⇒ H_{ij} には計 T^2 個の 3×3 ブロックがあるが, 4 つしか埋まらない
 - 行列 H は, 全エッジにおける H_{ij} の和
⇒ エッジ数 $|\mathcal{C}|$ は, ノード数 $T = t + 1$ より少し多い程度
⇒ 行列 H のうち, $4|\mathcal{C}|$ ブロックが埋まるが, 総数 T^2 より少ない
⇒ 行列 H もやはり **スパース**
 - $H_{[ji]} = H_{[ij]}^\top$ であるから, $H_{[ji]}$ は計算しなくてよい
 - さらに, 行列 H は **対称行列** であるから, 上三角部分だけ保持すればよい
 - 情報行列 Ω_0 は, 対角成分を十分大きな値とする
⇒ 初期姿勢 x_0 を \hat{x}_0 に固定

6 ポーズ調整のまとめ

- ポーズ調整の入出力
- ポーズ調整のアルゴリズム
- 1/4: 方程式の係数の計算
- 2/4: 方程式を解く
- 3/4: 解の更新
- 4/4: 収束判定
- ポーズ調整のためのライブラリ

ポーズ調整のまとめ

- ポーズ調整のアルゴリズム (2/4: 方程式を解く) [5]
 - 係数 H, b から構成した方程式を, Δx について解く

$$H\Delta x = -b$$

- 行列 H はスパース
 - ⇒ 疎行列コレスキー分解や, 共役勾配法を利用して解く
- コレスキー分解では, 並べ替え (Ordering) アルゴリズムを使う [11] [7]
- コレスキー分解は, 正定値対称行列 S を $S = LL^T$ のように分解
 - ⇒ コレスキー因子 L には, 新たに非ゼロ成分が生じてしまう
 - ⇒ 非ゼロ成分が増えると, 計算時間とストレージコストが増加する
- 分解を行う前に, 行列 S の行と列を適切に並べ替えておく
 - ⇒ 新たに発生する非ゼロ要素の数を, 抑えることが可能

ポーズ調整のまとめ

- ポーズ調整のアルゴリズム (2/4: 方程式を解く) [5]
 - 係数 H, b から構成した方程式を, Δx について解く

$$H\Delta x = -b$$

- 線形代数ライブラリ Eigen の場合 (C++)
 - ⇒ 疎行列 H の格納: `SparseMatrix, std::vector<Triplet>`
 - ⇒ 疎行列コレスキー分解: `SimplicialLDLT<SparseMatrix>`
 - ⇒ 共役勾配法: `ConjugateGradient<SparseMatrix>`
- Eigen の疎行列コレスキー分解では, AMD で並べ替えを行う
 - ⇒ AMD は, Approximate Minimum Degree の略
 - ⇒ 最小次数並べ替えアルゴリズム, グラフ理論に基づく手法

6 ポーズ調整のまとめ

- ポーズ調整の入出力
- ポーズ調整のアルゴリズム
- 1/4: 方程式の係数の計算
- 2/4: 方程式を解く
- 3/4: 解の更新
- 4/4: 収束判定
- ポーズ調整のためのライブラリ

ポーズ調整のまとめ

- ポーズ調整のアルゴリズム (3/4: 姿勢の更新) [5]
 - 方程式の解 Δx^* を使って, 全ノードの姿勢 x を更新する

$$x^* = \check{x} + \Delta x^*$$

- この際, $x^* = x_0^*, x_1^*, \dots, x_t^*$ の角度成分を $[-\pi, \pi)$ に丸める
 - ⇒ 特異点の問題を防ぐために必要 (3 次元の場合は少し複雑)
 - ⇒ 詳細は文献 [5] を参照
- 角度成分に関しては, 2π を足しても同じ意味をもつ
 - ⇒ 角度成分は位置 (x, y 成分) とは異なり, ユークリッド空間上にはない
 - ⇒ しかし, 最適化はユークリッド空間上で行っている
 - ⇒ 角度が境界 $(-\pi, \pi)$ をまたぐと破綻する

6 ポーズ調整のまとめ

- ポーズ調整の入出力
- ポーズ調整のアルゴリズム
- 1/4: 方程式の係数の計算
- 2/4: 方程式を解く
- 3/4: 解の更新
- 4/4: 収束判定
- ポーズ調整のためのライブラリ

ポーズ調整のまとめ

- ポーズ調整のアルゴリズム (4/4: 収束判定) [5]
 - 更新された姿勢 $x^* = \check{x} + \Delta x^*$ を使って, 目的関数 $F(x^*)$ を計算
 $\Rightarrow F(x^*) - F(\check{x}) < \varepsilon$ であれば, **収束判定**
 - そうでなければ, 初期値 \check{x} を x^* に設定して, 再度やり直す

6 ポーズ調整のまとめ

- ポーズ調整の入出力
- ポーズ調整のアルゴリズム
- 1/4: 方程式の係数の計算
- 2/4: 方程式を解く
- 3/4: 解の更新
- 4/4: 収束判定
- ポーズ調整のためのライブラリ

ポーズ調整のまとめ

- ポーズ調整のためのライブラリ

- g2o: A General Framework for Graph Optimization [8]
 - ⇒ グラフ最適化のためのライブラリ (C++)
 - ⇒ <https://github.com/RainerKuemmerle/g2o>
- p2o: Single header 2D/3D graph-based SLAM library
 - ⇒ 文献 [16] で使われているシングルヘッダライブラリ (C++)
 - ⇒ 短く読みやすいので, 参考になった
 - ⇒ <https://github.com/furo-org/p2o>

目次

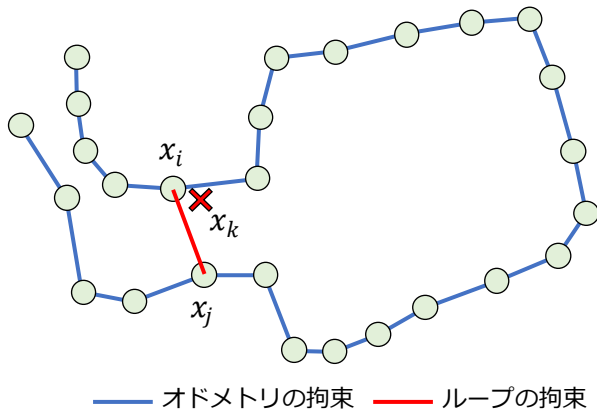
- ① イントロダクション
- ② グラフベース SLAM とは
- ③ グラフベース SLAM の定式化
- ④ ポーズグラフの構築
- ⑤ ポーズ調整の解法
- ⑥ ポーズ調整のまとめ
- ⑦ 細かな話題**
- ⑧ まとめ

7 細かな話題

- ループ辺の必要性
- ヤコビ行列の計算例
- レーベンバーグ・マーカート法の利用
- 外れ値への対処

細かな話題 (1)

- ループ辺とオドメトリ辺
 - 今回の場合, ループ辺がないと, ポーズ調整ができない



細かな話題 (1)

- ループ辺の必要性

- ループ辺がないと、ポーズ調整ができない
- ポーズ調整は、各エッジについて残差を最小化しようとする

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$$

- z_{ij} は、ノード i と j の相対姿勢 (拘束)
⇒ 観測 (スキャンマッチングやループ検出) によって得られる
- $\hat{z}_{ij}(x_i, x_j)$ は、最適化によって調整される相対姿勢 (x_i, x_j は変数)
⇒ ポーズグラフのノードの姿勢 x_i, x_j を調整
⇒ $\hat{z}_{ij}(x_i, x_j) = x_j \ominus x_i$ を、観測 z_{ij} に近づける

細かな話題 (1)

- ループ辺の必要性

- ポーズ調整は, 各エッジについて残差を最小化しようとする

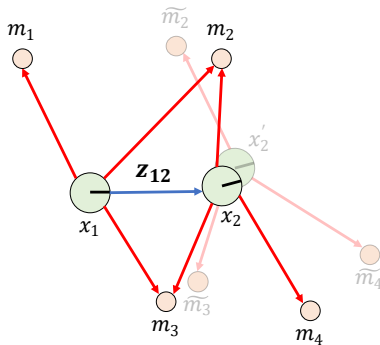
$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$$

- ポーズ調整では, 初期値 \check{x}_i, \check{x}_j のもとで上記を計算する
 - ⇒ \check{x}_i, \check{x}_j には, 逐次処理により求めたノードの姿勢を利用
 - ⇒ $e_{ij}(\check{x}_i, \check{x}_j) = z_{ij} - \hat{z}_{ij}(\check{x}_i, \check{x}_j)$ を計算する
- エッジ (i, j) がオドメトリ辺であるとする
 - ⇒ 隣接する時刻におけるノード i, j の間に張られている
- オドメトリ辺の場合, 残差 $e_{ij}(\check{x}_i, \check{x}_j)$ が 0 である

細かな話題 (1)

- ループ辺の必要性

- 下図の場合, 2つのノードの姿勢 (初期値) は $\check{x}_1 = x_1, \check{x}_2 = x_2$
 \Rightarrow エッジの観測 z_{12} は, ノード x_1, x_2 の相対姿勢 ($z_{12} = x_2 \ominus x_1$)
 $\Rightarrow z_{12}$ は, 初期値から求めた $\hat{z}_{12}(\check{x}_1, \check{x}_2) = \check{x}_2 \ominus \check{x}_1$ そのもの!



細かな話題 (1)

- ループ辺の必要性

- エッジ (i, j) がオドメトリ辺の場合, 残差 $e_{ij}(\check{x}_i, \check{x}_j)$ は 0 である
- グラフのエッジが全てオドメトリ辺であれば, ベクトル b が 0 になる

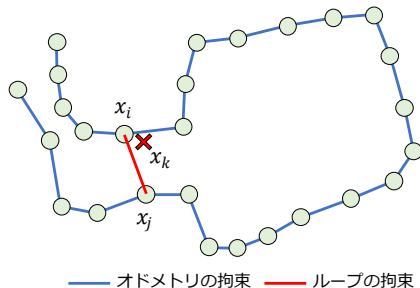
$$\begin{aligned} b &= \sum_{(i,j) \in \mathcal{C}} b_{ij} \\ &= \sum_{(i,j) \in \mathcal{C}} \left[\cdots e_{ij}(\check{x}_i, \check{x}_j)^\top \Omega_{ij} J_i \cdots e_{ij}(\check{x}_i, \check{x}_j)^\top \Omega_{ij} J_j \cdots \right] \end{aligned}$$

- 方程式 $H\Delta x = -b$ の右辺が 0 であるから, Δx も 0 となる
⇒ ノードの姿勢 $x_{0:t}^*$ が, 初期値 $\check{x}_{0:t}$ のままである
⇒ ポーズ調整の前後で, 全ノードの姿勢 $x_{0:t}$ が変化しない
- ポーズグラフにループ辺が 1 つもなければ, ポーズ調整ができない
⇐ 直感的には明らか

細かな話題 (1)

- ループ辺の必要性

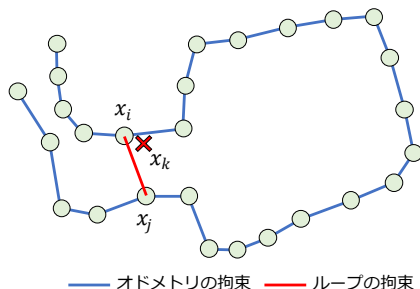
- エッジ (i, j) がループ辺の場合について考える
- $\hat{z}_{ij}(x_i, x_j)$ は, 2つのノード x_i, x_j から計算される相対姿勢
 - $\Rightarrow x_i$ と x_j は, 逐次実行により求められる
 - $\Rightarrow x_i$ から長い時間を経て x_j に辿り着く
 - $\Rightarrow x_j$ は累積誤差の影響を受けて, 実際の位置から外れている



細かな話題 (1)

- ループ辺の必要性

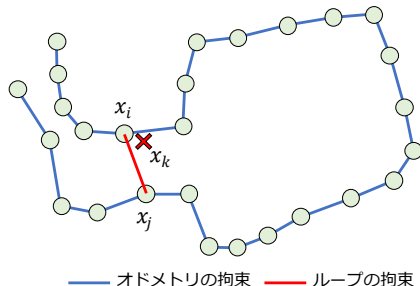
- $\hat{z}_{ij}(x_i, x_j)$ は, 2つのノード x_i, x_j から計算される相対姿勢
⇒ x_j は累積誤差の影響を受けて, 実際の位置から外れている
- z_{ij} は観測, 従って, ループ検出により得られる相対姿勢
⇒ z_{ij} は, \hat{z}_{ij} と比べれば, 比較的正確である



細かな話題 (1)

- ループ辺の必要性

- 観測 z_{ij} は x_j と x_k の差であり, \hat{z}_{ij} は x_i と x_j の差 (下図)
 - $\Rightarrow z_{ij}$ と \hat{z}_{ij} は, 通常大きく異なる
 - \Rightarrow ループ辺の場合は, 残差 $e_{ij}(x_i, x_j)$ が 0 にならない
 - $\Rightarrow b \neq 0, \Delta x \neq 0$ となって, ポーズ調整が機能する
- ポーズ調整を行うためには, ループ辺が必要である



7 細かな話題

- ループ辺の必要性
- ヤコビ行列の計算例
- レーベンバーグ・マーカート法の利用
- 外れ値への対処

細かな話題 (2)

- ポーズ調整ではヤコビ行列を計算する [5, 7]
 - 各エッジ (i, j) について, 2 つのヤコビ行列 J_i, J_j を計算する
⇐ 初期値 $x_i = \check{x}_i, x_j = \check{x}_j$ のもとでの, 残差 $e_{ij}(x_i, x_j)$ の偏微分

$$J_i = \left. \frac{\partial e_{ij}(x_i, x_j)}{\partial x_i} \right|_{x_i = \check{x}_i, x_j = \check{x}_j}$$
$$J_j = \left. \frac{\partial e_{ij}(x_i, x_j)}{\partial x_j} \right|_{x_i = \check{x}_i, x_j = \check{x}_j}$$

- $e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$ である
 - ⇒ z_{ij} は定数項なので, 無視できる
 - ⇒ $\hat{z}_{ij}(x_i, x_j)$ の x_i, x_j による偏微分を考えればよい
- $\hat{z}_{ij}(x_i, x_j) = x_j \ominus x_i$ は, x_i と x_j の相対姿勢を求める関数

細かな話題 (2)

- 相対姿勢を求める関数 [16, 5, 7]

- $\hat{z}_{ij}(x_i, x_j) = x_j \ominus x_i$ は, x_i と x_j の相対姿勢を求める関数
- $x_i = [\xi_i^x, \xi_i^y, \xi_i^\theta]^\top$, $x_j = [\xi_j^x, \xi_j^y, \xi_j^\theta]^\top$ とすると,

$$\begin{aligned} & \hat{z}_{ij}(x_i, x_j) \\ = & x_j \ominus x_i \\ = & \begin{bmatrix} \cos \xi_i^\theta & \sin \xi_i^\theta & 0 \\ -\sin \xi_i^\theta & \cos \xi_i^\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \xi_j^x - \xi_i^x \\ \xi_j^y - \xi_i^y \\ \xi_j^\theta - \xi_i^\theta \end{bmatrix} \\ = & \begin{bmatrix} \cos \xi_i^\theta (\xi_j^x - \xi_i^x) + \sin \xi_i^\theta (\xi_j^y - \xi_i^y) \\ -\sin \xi_i^\theta (\xi_j^x - \xi_i^x) + \cos \xi_i^\theta (\xi_j^y - \xi_i^y) \\ \xi_j^\theta - \xi_i^\theta \end{bmatrix} \end{aligned}$$

細かな話題 (2)

- ヤコビ行列の計算

- ヤコビ行列 J_i は次のようになる [5, 7]

$$\begin{aligned} J_i &= \frac{\partial e_{ij}(x_i, x_j)}{\partial x_i} = \frac{\partial}{\partial x_i} (z_{ij} - \hat{z}_{ij}(x_i, x_j)) \\ &= - \frac{\partial \hat{z}_{ij}(x_i, x_j)}{\partial x_i} \\ &= \begin{bmatrix} \cos \xi_i^\theta & \sin \xi_i^\theta & \sin \xi_i^\theta (\xi_j^x - \xi_i^x) - \cos \xi_i^\theta (\xi_j^y - \xi_i^y) \\ -\sin \xi_i^\theta & \cos \xi_i^\theta & \cos \xi_i^\theta (\xi_j^x - \xi_i^x) + \sin \xi_i^\theta (\xi_j^y - \xi_i^y) \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} R(\xi_i^\theta)^\top & -\frac{\partial R(\xi_i^\theta)^\top}{\partial \xi_i^\theta} (t_j - t_i) \\ 0 & 1 \end{bmatrix} \end{aligned}$$

細かな話題 (2)

- ヤコビ行列の計算

- 3×3 のヤコビ行列 J_i は次のようになる [5, 7]

$$\begin{aligned} J_i &= \begin{bmatrix} \cos \xi_i^\theta & \sin \xi_i^\theta & \sin \xi_i^\theta (\xi_j^x - \xi_i^x) - \cos \xi_i^\theta (\xi_j^y - \xi_i^y) \\ -\sin \xi_i^\theta & \cos \xi_i^\theta & \cos \xi_i^\theta (\xi_j^x - \xi_i^x) + \sin \xi_i^\theta (\xi_j^y - \xi_i^y) \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} R(\xi_i^\theta)^\top & -\frac{\partial R(\xi_i^\theta)^\top}{\partial \xi_i^\theta} (t_j - t_i) \\ 0^\top & 1 \end{bmatrix} \end{aligned}$$

- 但し, $t_i = [\xi_i^x, \xi_i^y]^\top$, $t_j = [\xi_j^x, \xi_j^y]^\top$
- $R(\xi_i^\theta)$ は, ξ_i^θ により定まる, 2×2 の回転行列

$$R(\xi_i^\theta) = \begin{bmatrix} \cos \xi_i^\theta & -\sin \xi_i^\theta \\ \sin \xi_i^\theta & \cos \xi_i^\theta \end{bmatrix} \quad (60)$$

細かな話題 (2)

- ヤコビ行列の計算

- ヤコビ行列 J_j も同様に求められる [5, 7]

$$\begin{aligned} J_j &= \frac{\partial e_{ij}(x_i, x_j)}{\partial x_j} \\ &= - \frac{\partial \hat{z}_{ij}(x_i, x_j)}{\partial x_j} \\ &= \begin{bmatrix} -\cos \xi_i^\theta & -\sin \xi_i^\theta & 0 \\ \sin \xi_i^\theta & -\cos \xi_i^\theta & 0 \\ 0 & 0 & -1 \end{bmatrix} \\ &= \begin{bmatrix} -R(\xi_i^\theta)^\top & 0 \\ 0^\top & -1 \end{bmatrix} \end{aligned} \tag{61}$$

- これらを使って、ポーズ調整の計算を進める (2 次元の場合)
- 3 次元の場合の計算は [5] を参照

7 細かな話題

- ループ辺の必要性
- ヤコビ行列の計算例
- レーベンバーグ・マーカート法の利用
- 外れ値への対処

細かな話題 (3)

- レーベンバーグ・マーカート法の利用 [7, 15, 14]
 - ガウス・ニュートン法によるポーズ調整では, 次の方程式を解いた

$$H\Delta x = -b$$

- レーベンバーグ・マーカート法を用いる場合は, 次の方程式を解く [7]

$$(H + \lambda I) \Delta x = -b \quad (62)$$

- λI は, H の対角成分を用いて, $\lambda \text{diag } H$ としてもよい
- $\lambda \geq 0$ は, ガウス・ニュートン法と最急降下法を調節する
 - $\Rightarrow \lambda = 0$ のとき, **ガウス・ニュートン法**に一致
 - $\Rightarrow \lambda \rightarrow \infty$ のとき, **最急降下法**に一致

細かな話題 (3)

- レーベンバーグ・マーカート法の利用 [7, 15, 14]
 - $\lambda \geq 0$ は, ガウス・ニュートン法と最急降下法を調節する
 - $\Rightarrow \lambda = 0$ のとき, **ガウス・ニュートン法**に一致
 - $\Rightarrow \lambda \rightarrow \infty$ のとき, **最急降下法**に一致
 - ガウス・ニュートン法は, 目的関数のヘッセ行列を近似する
 - \Rightarrow この近似は, 解の近傍でのみ有効である
 - 解から離れているとき, 勾配法を用いて大まかな探索を行う
 - \Rightarrow 解にある程度近づいたら, ガウス・ニュートン法に切り替える
 - \Rightarrow レーベンバーグ・マーカート法は, この切り替えを λ で行う

細かな話題 (3)

- レーベンバーグ・マーカート法の利用 [7, 15, 14]
 - レーベンバーグ・マーカート法を用いる場合は, 次の方程式を解く

$$(H + \lambda I) \Delta x = -b$$

- 目的関数 $F(x_{0:t})$ が増加した場合, λ を大きくする ($\lambda \leftarrow 10\lambda$)
 $\Leftrightarrow F(x_{0:t})$ が減少した場合は, λ を小さくする ($\lambda \leftarrow \frac{1}{10}\lambda$)
- 新たなループ辺が追加されると, $F(x_{0:t})$ が増加する
 - \Rightarrow 現在の姿勢 $x_{0:t}$ が, 最適解 $x_{0:t}^*$ から離れている状態
 - $\Rightarrow \lambda$ を大きくして最急降下法に切り替え, 大まかな探索を行う
 - \Rightarrow 最適解にある程度近いところまで到達
 - $\Rightarrow \lambda$ を小さくして, ガウス・ニュートン法に切り替える
 - \Rightarrow ノードの姿勢 $x_{0:t}$ を正しく修正できる

7 細かな話題

- ループ辺の必要性
- ヤコビ行列の計算例
- レーベンバーグ・マーカート法の利用
- 外れ値への対処

細かな話題 (4)

- 外れ値への対処 [6]

- ポーズ調整は, 次の目的関数 $F(x_{0:t})$ の最小化

$$\begin{aligned} F(x_{0:t}) &= \sum_{(i,j) \in \mathcal{C}} F_{ij}(x_i, x_j) \\ &= \sum_{(i,j) \in \mathcal{C}} (e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j)) \end{aligned}$$

- 誤ったループ検出 (外れ値) は, アルゴリズムに悪影響を与える
 - ⇒ ループ辺 (i, j) が誤って追加されたとする
 - ⇒ エッジの残差 $e_{ij}(x_i, x_j)$ が大きくなる
 - ⇒ 目的関数に, 過剰に大きな二乗誤差 $F_{ij}(x_i, x_j)$ が足される

細かな話題 (4)

- 外れ値への対処 [6]

- ポーズ調整は, 次の目的関数 $F(x_{0:t})$ の最小化

$$F(x_{0:t}) = \sum_{(i,j) \in \mathcal{C}} (e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j))$$

- 外れ値に敏感ではない関数 $\rho(\cdot)$ を使う

⇒ $\rho(\cdot)$ は, 二乗誤差 $F_{ij}(x_i, x_j)$ よりは敏感でない

⇒ 外れ値による影響を低減

- $\rho(\cdot)$ を使って, 目的関数を次のように記述

$$F(x_{0:t}) = \sum_{(i,j) \in \mathcal{C}} \rho(e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j)) \quad (63)$$

細かな話題 (4)

- 外れ値への対処 [6]

- $\rho(\cdot)$ を使って, 目的関数を次のように記述 (ロバスト推定)

$$F(x_{0:t}) = \sum_{(i,j) \in \mathcal{C}} \rho(e_{ij}(x_i, x_j)^\top \Omega_{ij} e_{ij}(x_i, x_j))$$

- 文献 [6] では, Huber 損失関数を利用

$$\rho(s) = \begin{cases} \frac{1}{2}s^2 & |s| \leq \delta \\ \delta(|s| - \frac{1}{2}\delta) & \text{Otherwise} \end{cases} \quad (64)$$

- ヤコビ行列 J や更新量 Δx を適切にスケーリングする
- 詳しく知らないので, 文献 [6, 13] を参照
- 外れ値の対処法として, Dynamic Covariance Scaling [1] も参照

目次

- ① イントロダクション
- ② グラフベース SLAM とは
- ③ グラフベース SLAM の定式化
- ④ ポーズグラフの構築
- ⑤ ポーズ調整の解法
- ⑥ ポーズ調整のまとめ
- ⑦ 細かな話題
- ⑧ まとめ

- このスライドのまとめ
 - SLAM の概要
 - グラフベース SLAM の定式化
 - ポーズグラフの中身と構築方法
 - ポーズ調整のアルゴリズム
 - 細かな話
 - ヤコビ行列の計算例
 - レーベンバーグ・マーカート法
 - ロバスト推定 (M 推定)
- 初心者なので、間違っている部分があるかもしれません
- ご清聴有難うございました

- [1] Pratik Agarwal, Gian Diego Tipaldi, Luciano Spinello, Cyrill Stachniss, and Wolfram Burgard.
Robust map optimization using dynamic covariance scaling.
In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 62–69, 2013.
- [2] Ola Bengtsson and Albert-Jan Baerveldt.
Localization in changing environments - estimation of a covariance matrix for the idc algorithm.
In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1931–1937 vol.4, 2001.
- [3] Peter Biber and Wolfgang Straßer.
The Normal Distributions Transform: A New Approach to Laser Scan Matching.
In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2743–2748 vol.3, 2003.
- [4] Andrea Censi.
An accurate closed-form estimate of icp's covariance.
In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 3167–3172, 2007.

- [5] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard.
A Tutorial on Graph-Based SLAM.
IEEE Transactions on Intelligent Transportation Systems Magazine, 2(4):31–43, December 2010.
- [6] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor.
Real-Time Loop Closure in 2D LIDAR SLAM.
In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 1271–1278, 2016.
- [7] Kurt Konolige, Giorgio Grisetti, Rainer Kümmerle, Wolfram Burgard, Benson Limketkai, and Regis Vincent.
Efficient Sparse Pose Adjustment for 2D mapping.
In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 22–29, 2010.
- [8] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard.
g2o: A General Framework for Graph Optimization.
In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 3607–3613, 2011.

- [9] Luis Montesano, Javier Minguez, and Luis Montano.
Probabilistic scan matching for motion estimation in unstructured environments.
In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3499–3504, 2005.
- [10] Edwin B. Olson.
Real-Time Correlative Scan Matching.
In Proceedings of the IEEE International Conference on Robotics and Automation, pages 4387–4393, 2009.
- [11] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery.
Numerical Recipes 3rd Edition: The Art of Scientific Computing.
Cambridge University Press, 2007.
- [12] Sebastian Thrun, Wolfram Burgard, and Dieter Fox.
Probabilistic Robotics.
MIT Press, 2005.
- [13] Bill Triggs, Philip Mclauchlan, Richard Hartley, and Andrew Fitzgibbon.
Bundle Adjustment - A Modern Synthesis.
In Proceedings of the International Workshop on Vision Algorithms: Theory and Practice, pages 298–372, September 1999.

参考文献

- [14] 金森 敬文, 鈴木 大慈, 竹内 一郎, and 佐藤 一誠.
機械学習プロフェッショナルシリーズ機械学習のための連続最適化.
講談社, 2016.
- [15] 金谷 健一.
これなら分かる最適化数学 –基礎原理から計算手法まで–.
共立出版, 2005.
- [16] 友納 正裕.
SLAM 入門 –ロボットの自己位置推定と地図構築の技術–.
オーム社, 2018.
- [17] 上田 隆一.
詳解 確率ロボティクス *Python* による基礎アルゴリズムの実装.
講談社, 2019.