自己位置の連続性を考慮した SLAM 処理のコンテナ間引継ぎ手法の提案

久保田 幸士郎[1]　菅谷 みどり[1]

概要：自律移動ロボットにおいて，SLAM（Simultaneous Localization and Mapping）は自律移動に必要なデータ算出を行う技術として普及している．一方，本技術は多数のセンサから得られるデータを元に継続的に計算処理を行うため，計算負荷が一般的に高くなる．この負荷を軽減するため，近年では通信技術の発展もあり，SLAM 処理を外部サーバへオフロードする方式が検討されている．こうしたシステムでは，サーバの負荷変動などに対応するため，実行中の SLAM 処理を別の実行環境へ引き継げることが望ましい．しかし，SLAM は地図や自己位置を継続的に更新するステートフルな処理であり，保存した内部状態を切替先へ引き継ぐだけでは，状態保存後のロボットの移動を切替先の初期姿勢に反映できず，切替前後の自己位置推定結果が不連続になる場合がある．このような不連続が生じると，実際のロボットの移動とは対応しない位置の急変が地図上に生じ，自己位置情報を利用する経路計画や走行制御に影響を及ぼす恐れがある．本研究ではこの不連続の抑制を目的とし，実現のためにコンテナ間引継ぎ処理中のロボット移動量を odometry に基づいて補正する処理を SLAM 処理のコンテナ間引継ぎシステムに組み合わせる．提案手法では，状態保存後のロボットの移動量を odometry から算出し，切替先で開始する SLAM 処理の初期姿勢に反映する．また，ROS 2 と Cartographer を用いた SLAM 処理のコンテナ間引継ぎシステムを実装した．評価実験では，姿勢補正の有無による切替前後の自己位置出力の連続性を比較し，提案手法によって引継ぎ時の位置不連続を最大約 67.6%抑制することにより，提案手法の有効性を示した．

キーワード：Simultaneous Localization and Mapping，オフローディング，分散システム

1. 背景

近年，物流倉庫における物資運搬[1]や，施設内での清掃[2]など，複数台の自律移動ロボットを用いたサービスが普及している．これらの自律移動ロボットでは，周囲環境の地図生成と地図上における自己位置推定を同時に行う Simultaneous Localization and Mapping（SLAM）[3]が普及している. SLAM により得られる地図および自己位置情報は，自律移動ロボットにおいて，移動する際の経路計画や走行制御を行うための基盤となる．

SLAM は，自己位置と地図を更新するためにロボットに装着された LiDAR（Light Detection And Ranging，レーザー光による計測装置）や，カメラ，IMU（Inertial Measurement Unit，慣性計測装置），車輪オドメトリ等のセンサから得られるデータを継続的に処理する．そのため，CPU やメモリ等の計算資源を継続的に消費する．一方，屋内向け自律移動ロボットにおいては，重量やサイズ，消費電力量の制限から SLAM に必要とされる計算量を十分に賄う CPU やメモリを搭載した高性能計算機を用いることが困難であることが多い．こういったロボットの限られた計算資源で SLAM を行う場合, ロボット上の CPU やメモリを継続的に占有してしまう．その結果，経路計画や認識処理など，SLAM 以外の処理周期や応答性能へ影響を与えることが課題となっている．

2. 関連研究

この課題に対し，近年の無線通信技術の発展もあり，SLAM をロボット外部のエッジサーバやクラウドサーバへオフロードする方式が負荷軽減手法の一つとして注目されている．SLAM のオフロードによる計算負荷の解決手法に関して，Huang らの提案がある[4]．Huang らは，SLAM 処理のオフロードに伴う処理遅延の課題に着目した．提案では，複数のロボットから取得したセンサデータをエッジサーバで処理し，ロボットごとの地図を生成・統合する．さらに, 各エッジサーバで統合した地図をクラウドへ送信し，全体地図を生成する．このように段階的な地図統合処理を行うことで，クラウドへ送信するデータを削減し，システムの総実行時間の遅延を削減した．また，ロボットのデータをエッジサーバへ適応的に割り当てることで，エッジサーバ間の負荷分散も考慮している．

一方，ロボットの増加などによって SLAM 処理を実行するサーバの負荷が変化する場合，処理の実行先を変更できる構成も考えられる．しかし，SLAM は地図やロボット姿勢などの情報を持つステートフルな処理であるため，実行先を変更して処理を継続するには，これらのステートを切替先で利用できるようにする仕組みが不可欠となる．しかし，Huang らの提案は，ロボットのデータ割当てによる負荷分散を扱っている一方, 実行中の SLAM が保持する状態を別のサーバへ引き継ぎ，処理を継続する機構については示されていない．

一方 Sarker らは，スマートエッジゲートウェイを用いた SLAM のオフロード方式を提案した[5]. この方式では, Fog 層の分散ストレージを介して地図などのデータを共有することで，ゲートウェイの障害発生時にも別のゲートウェイへ接続して SLAM 処理を継続可能とする構成を示している．しかし，複数ゲートウェイ間での負荷分散や，実行中の SLAM 状態の引継ぎについて, 具体的な実装および評価は示されていない．

一方で，SLAM という一つの処理を複数サーバに分散す

1 芝浦工業大学
Shibaura Institute of Technology

ⓒ2026 Information Processing Society of Japan

ることで，システム全体の負荷軽減を狙う研究も行われている．Zhang らはクラウド上の複数計算ホストにマルチロボット Visual SLAM を分散し，ホスト間で SLAM データを共有する手法を提案している[6]．この提案は，複数のロボットが同じ地図領域を見ている場合，それらのロボットを群としてまとめ，こうしてまとめられた各ロボット群を複数ホストへ分割する．これにより，通信コストを抑えながら協調 Visual SLAM の並列化に成功している．

また，Huang らは，複数ロボットが生成した地図を，複数のエッジサーバで効率よく統合するため，地図同士の重複度合を考慮して処理をエッジサーバへ割り当てる手法を提案した[7]．これにより，地図統合に必要な通信や必要な処理数を減らし，サーバ自体の負荷を軽減した．

しかし，これらの方式は SLAM の分割や地図統合タスクの割り当てを最適化することで，複数サーバを利用した協調 SLAM のスケーラビリティや処理性能を向上させるものである．一方，SLAM をオフロードする外部のサーバを利用する場合は，他のアプリケーションとの資源共有や，負荷変動，サーバ障害が発生する可能性がある．このような状況では，特定サーバで実行中の SLAM が負荷集中の原因となる場合もあるため，SLAM の処理効率だけでなく，実行中の SLAM をサーバ運用上の単位として別の実行環境へ再配置できることが重要となる．しかし，上述の通り SLAM は地図，ロボット姿勢，サブマップ等の状態を継続的に更新するステートフルな処理である．そのため，SLAM の実行先を変更する場合，単にセンサデータの転送先を切り替えるだけでは，切替元で生成された地図および自己位置推定状態を継続できない．

そこで，著者らの従来実装では，各コンテナ内の SLAM アルゴリズム（Cartographer）が保持する SLAM 状態を pbstream 形式でデータベースへ保存し，切替先コンテナで読み込むことにより，切替元で生成された地図を引き継いで SLAM を再開する方式を実装した[8]．図 1 に従来実装のシステム構成を示す．

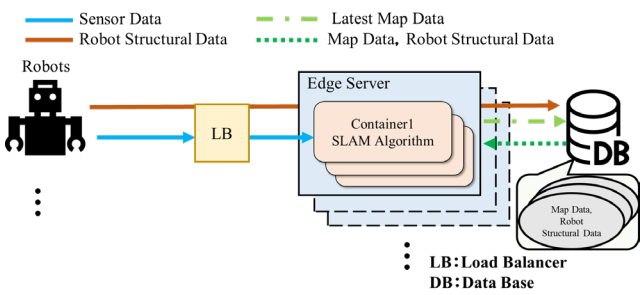

図 1　従来実装のシステム構成[8]

本方式により，切替先で地図を最初から生成し直すことなく SLAM を再開でき，SLAM をコンテナ単位で動かしつつコンテナ間における処理の切替を実現した．

一方，切替を行う前に pbstream として保存される状態は，状態を保存した時点のものである．システム側が切替を行う最中でもロボットは動き続ける場合，保存された状態が表すロボットの姿勢と，切替後に入力されるセンサデータが表す現在姿勢との間に差が生じる．その結果，切替先 Cartographer が出力する自己位置が，切り替え直前の切替元 Cartographer の出力に対して不連続になる場合がある．この不連続が，SLAM の処理結果を利用する自律移動制御にも影響を与える恐れがある．

これと似た事例に対し，Hofer らは，エッジ支援型 SLAM において，地図同期中に生じる通信遅延がロボットの位置追跡に与える影響を軽減するため，同期中にロボットが取得したカメラ画像を一時的に保持する循環型バッファを用いた手法を提案した[9]．これにより，地図同期後も位置追跡を継続できる信頼性を向上させた．このように，SLAM 状態の同期を伴うシステムでは，同期中のロボット移動を考慮することが重要となる．

3. 目的・提案

3.1 目的

本研究の目的は，コンテナ内で SLAM 処理を行う移動中のロボットがコンテナを切り替える際に，SLAM の出力する自己位置の不連続を抑制しつつ，切り替えを行う方法を提案する．自己位置の連続性とは，センサデータの転送先が切替元から切替先へ変更される境界において，ロボット自身の通常の移動量を除いた地図上の姿勢変化が小さいことを指す． SLAM 状態のコンテナ間における引継ぎと，状態保存後のロボットの移動量に基づく姿勢補正を組み合わせる本方式は，我々が知るかぎり，まだ実現されていない．

3.2 提案概要

本提案では，SLAM 状態の引継ぎと，状態保存後のロボットの移動量に基づく姿勢補正を組み合わせる．本方式では，ロボットのセンサデータを外部サーバ上の SLAM コンテナへ転送するオフロード環境を対象とし，SLAM の負荷に基づく切替先の選択そのものではなく，切替が決定された後に，Cartographer の状態とロボットの姿勢を別のコンテナに引き継ぐ処理を対象とする．

実現方法の概要を述べる．本方式では，切替元 Cartographer の SLAM 状態を pbstream 形式で Redis へ保存する．このとき，地図上におけるロボットの位置と向き（map 姿勢）と，odometry に基づく位置と向き（odom 姿勢）も併せて保存する．

Gateway はロボットから受信した odometry の最新値を保持する．切替先で SLAM 状態を読み込んだ後，保存時の odom 姿勢と Gateway が保持する最新の odom 姿勢から，保存後のロボットの相対移動量を算出する．この移動量を保存時の map 姿勢に合成することで，姿勢補正に使用した最

ⓒ2026 Information Processing Society of Japan

新 odometry の時刻における地図上のロボット姿勢を推定する．

切替先では，引き継いだ SLAM 状態を参照しながら，算出した姿勢補正に基づいて新しい trajectory を開始する．その後，Gateway のセンサデータ転送先を切替先コンテナへ変更する．これにより，状態保存から補正処理までに生じたロボットの移動を考慮し，切替前後の自己位置出力の不連続を抑制する．

3.3 システム構成

図 2 に提案システムの構成を示す．

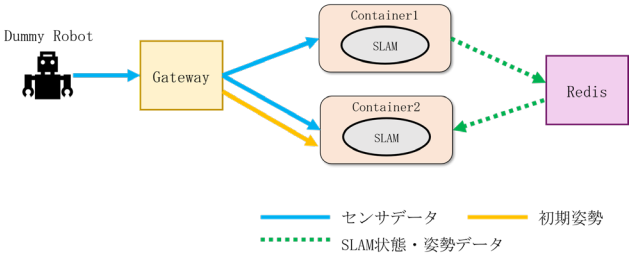

図 2　提案システムの構成

提案システムは，Dummy Robot，Gateway，Redis，切替元 SLAM（図中コンテナ 1），切替先 SLAM（図中コンテナ 2）の計 5 コンテナから構成される．今回の実装では，これらを同一の計算機上で動作させた．

Dummy Robot コンテナは rosbag の再生によって得られる scan，odom，joint_states の各トピックのデータを HTTP で Gateway へ送信する．Gateway は受信したセンサデータを，現在の転送先として選択されている SLAM コンテナへ転送する．切替元と切替先の SLAM コンテナには異なる ROS_DOMAIN_ID を設定し，ROS 2 の通信を分離した．切替の実行制御は Docker ホスト上のスクリプトから行う．各 SLAM コンテナには ROS 2 サービスを呼び出すための制御プロセスを常駐させ，ホスト側の制御プログラムから状態保存・読込みや trajectory の開始を実行する．切替時には，Redis を介して SLAM 状態と姿勢情報を引継ぎ，切替先で新 trajectory の開始を確認した後，Gateway がセンサデータの転送先を変更する．

3.4 Cartographer の状態引継ぎ機能

本実装では，2 次元 SLAM アルゴリズムとして Cartographer を用いた．Cartographer は，センサデータからロボットの自己位置を推定するとともに，周辺環境を部分的な地図（submap）として生成する．また，推定された位置関係をグラフとして管理・最適化することで，地図と自己位置推定結果を調整する[10]．Cartographer では，移動に伴って蓄積される姿勢や submap などを trajectory という単位で管理する．

Cartographer の標準的な状態保存機能では，SLAM 状態を pbstream 形式として保存することができる．本実装では，コンテナ間で状態を共有するため，実行中の Cartographer から SLAM 状態を Redis へ保存する ROS 2 サービスと，Redis から取得した状態を Cartographer へ読み込む ROS 2 サービスを追加した．状態保存時には，map 座標系および odom 座標系におけるロボット姿勢を，共通の TF 時刻で取得して Redis へ保存する．切替先では，pbstream から復元した既存 trajectory を FROZEN 状態として保持する．その後，この既存 trajectory を参照して新しい trajectory を開始する．初期姿勢の指定には，Cartographer ROS が提供する StartTrajectory サービスの initial_pose および relative_to_trajectory_id を使用した．

3.5 Gateway および Redis

Gateway は Go 言語で実装した．Gateway は，Dummy Robot から送信されたセンサデータを受信し，選択中の SLAM コンテナへ転送する．また，受信した odometry から最新のロボット姿勢を保持し，姿勢補正要求に応じて補正姿勢を算出する．なお，ここで言う odometry とは，ロボットの移動量や姿勢を推定したデータを指し，odom はその推定姿勢を表現するための座標系である．

SLAM 状態および姿勢情報の共有には Redis[11]を使用した．Redis はインメモリ型のキーバリューストアであり，pbstream 形式の SLAM 状態をバイナリデータとして，map 姿勢および odom 姿勢を JSON 形式で保存する．これにより，切替元と切替先のコンテナが共通の状態情報を利用できる．

3.6 姿勢補正手法

本提案手法におけるロボット姿勢補正手法について説明する．まず，本実装ではロボットの姿勢を位置と向きを含む剛体変換として扱う[12]．座標系 B の姿勢を座標系 A からみた剛体変換を${}^{A}T_{B}$と表し，式 1 のように定義する．ここで，R は回転行列，$\boldsymbol{p}$は並進ベクトルを表す．なお，実装では位置 3 成分と回転クォータニオンを用いて計算する．

$$ {}^{A}T_{B} = \begin{bmatrix} R & p \\ 0^{T} & 1 \end{bmatrix} \tag{1} $$

ロボット姿勢の取得および座標間の変換には，ROS の TF を用いる[13]．本実装では，map，odom，base_footprint の 3 つの座標系を使用する．map は地図上の自己位置推定の基準となる座標系，odom は odometry に基づきロボットの移動を連続的に表す座標系である[14]．base_footprint は床面上におけるロボット本体の基準座標系である．

ここで，M，O，B はそれぞれ map，odom，base_footprint 座標系を示し，e は Export 処理時の姿勢取得時点，c は補正に使用する最新 odometry の時点を表す．姿勢補正には以下の 3 つの姿勢を用いる．

ⓒ2026 Information Processing Society of Japan

$$^{M}T_{B,e} \tag{2}$$

$$^{O}T_{B,e} \tag{3}$$

$$^{O}T_{B,c} \tag{4}$$

式 2 は Export 時点の map 座標系上のロボット姿勢を示し，式 3 は Export 時点の odom 座標系上のロボット姿勢を示す．式 4 は Gateway が保持する最新の odom 座標系上のロボット姿勢を示す．これらのうち式 3 と式 4 で示した odom 姿勢に剛体変換の逆変換と合成[12]を適用し，Export 時の姿勢取得時点から，Gateway が姿勢補正に使用した最新 odometry の時点までの相対移動量を求める．その算出式を式 5 に示す．

$$\Delta T_{\mathrm{odom}} = \left(^{O}T_{B,e}\right)^{-1} {}^{O}T_{B,c} \tag{5}$$

式 5 は，同一の odom 座標系で取得した 2 時点の姿勢から，状態保存時点のロボット姿勢を基準とする相対移動量を求めるものである．この相対移動量を，式 2 で示した状態保存時点の map 座標系上のロボット姿勢に合成することで，補正処理時点における map 座標系上のロボット姿勢を推定する．その算出式を式 6 に示す．

$$^{M}T_{B,c}^{\mathrm{corrected}} = {}^{M}T_{B,e}\,\Delta T_{\mathrm{odom}} \tag{6}$$

式 6 により，状態保存後のロボットの移動量を反映した map 座標系上の補正姿勢が得られる．

Cartographer の StartTrajectory サービスでは，既存 trajectory を基準とした相対初期姿勢を指定する[15]．本実装では，読み込んだ既存 trajectory の最初のノードの map 座標系上の姿勢を$^{M}T_{\mathrm{ref}}$とし，式 6 で算出した補正姿勢を相対初期姿勢へ変換する．その算出式を式 7 に示す．

$$T_{\mathrm{init}} = \left(^{M}T_{\mathrm{ref}}\right)^{-1} {}^{M}T_{B,c}^{\mathrm{corr}} \tag{7}$$

ここで，T_{init}は StartTrajectory に指定する相対初期姿勢を表す．この相対変換を initial_pose として指定し，relative_to_trajectory_id に既存 trajectory の ID を設定することで，新 trajectory を開始する．これにより，状態保存後のロボットの移動量を初期姿勢に反映し，切替前後における自己位置出力の不連続の抑制を図る．

3.7 引継ぎ処理手順

次に本提案システムにおける引継ぎ時の処理手順について述べる．まず引継ぎ処理のシーケンス図を以下に示す．

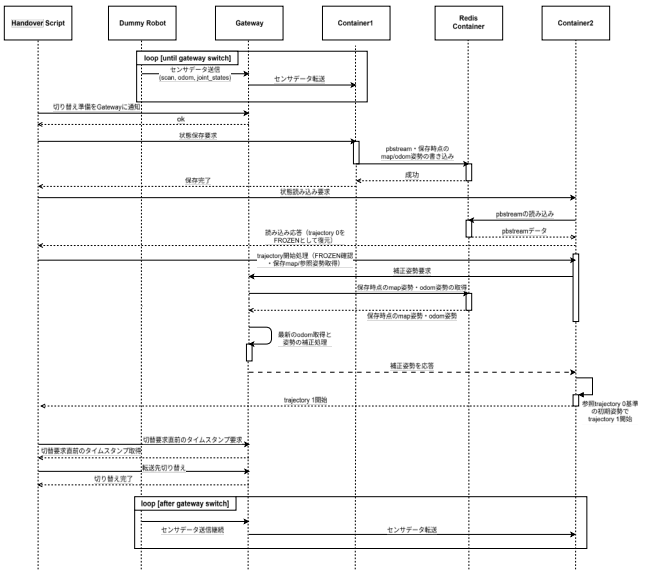

図 3　引継ぎ処理シーケンス図

前提として，Gateway，Redis，切替元（図中コンテナ 1）と切替先（コンテナ 2）での Cartographer とセンサデータを受け取るための HTTP サーバは起動済みで，コンテナ 1 では初期 trajectory を開始済みのものとする．

引継ぎ処理開始時，ホスト側の制御プログラムは Gateway へ切替準備を要求する．この段階では Gateway は切替元へのセンサデータの転送を継続する．続いて，切替元 Cartographer の SLAM 状態を pbstream 形式で Redis へ保存するとともに，Export 処理時に取得した map 姿勢および odom 姿勢を Redis へ保存する．

切替先では Redis から pbstream を読み込み，既存 trajectory が FROZEN 状態となり，参照するノードの姿勢を取得できることを確認する．その後，Gateway へ姿勢補正を要求し，得られた補正姿勢を既存 trajectory に対する相対初期姿勢へ変換する．この初期姿勢を用いて切替先 Cartographer の新 trajectory を開始する．

新 trajectory の開始後，trajectory の状態と必要なセンサトピックの購読状態を確認する．確認が完了すると，Gateway の転送先を切替元から切替先へ変更する．その後，新 trajectory において切替基準時刻より新しいセンサ時刻を持つノードが生成されたことを確認する．これにより，切替先で SLAM 処理が進行していることを確認する．それ以降は，切替先 Cartographer が受信したセンサデータを用いて SLAM を継続する．

4. 評価

4.1 評価目的と比較条件

本評価では，コンテナ間で SLAM を引き継ぐ際の自己位置出力の連続性について，切替先 trajectory の初期姿勢の与え方を変えた 2 条件を比較する．条件 A は状態保存時点の map 姿勢を初期値とする方式，条件 B は保存時の map 姿勢

ⓒ2026 Information Processing Society of Japan

に状態保存後の odometry による移動量を合成する提案方式である．両条件とも同じ方式で pbstream を引き継いで新しい trajectory を開始し，保存後の移動量を初期姿勢に反映するかどうかを比較する．

評価指標には，ロボットの移動を odometry で補償した後の位置残差および yaw 残差を用いる．切替直後と切替から 5，10，30 秒後を対象とする．

また，SLAM 状態の引継ぎに要する時間とその内訳を計測し，処理時間の大部分を占める処理を明らかにするとともに，条件 A と B の引継ぎ時間を比較する．

本評価が対象とするのは，切替元と切替先の Cartographer が出力する map 座標系上の姿勢の連続性である．物理空間における絶対位置精度および占有格子地図の形状整合性は対象としない．

4.2 実験環境

実験環境は図 2 と同じ 5 コンテナから構成され，すべて単一の計算機上で動作させた．コンテナ間通信には Docker の host network を使用し，OS は Ubuntu 24.04，ROS 2 は Jazzy とした．使用した計算機の性能を表 1 に示す．

表 1 使用した計算機の性能

	PC
CPU	Intel Core i5-8350U
メモリ(GiB)	40

TF は map→odom，odom→base_footprint および map→base_footprint を共通時刻で記録した．同一走行条件では共通の rosbag とハンドオーバー開始目標時刻を使用し，実際の切替時刻の小さな差を含む独立した反復試行として扱う．rosbag とは，ROS 2 トピックに流れるデータをレコードする機能であり，本評価では実際にロボットを動かして取得した rosbag レコード内のセンサデータを Gateway に転送するコンテナを作成することで，疑似的に動いているロボットからセンサデータが送られてくる環境を再現した．また，直進条件および旋回条件のそれぞれで，条件 A と B を各 10 回，合計 40 回の完了試行を分析した．

4.3 走行条件

本評価では，Dummy Robot コンテナから rosbag に記録されたセンサデータを Gateway へ送信し，scan および odom を用いて SLAM を実行した．使用した走行軌跡と生成地図を図 4 に示す．

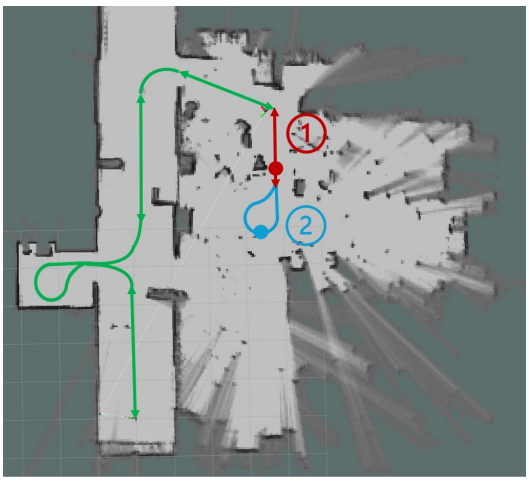

図 4 Dummy Robot コンテナで再生するレコードが最終的に作成した地図

図 4 の①は直進区間，②は旋回区間である．この区間中のそれぞれ点部分で引継ぎ処理を開始した．

4.4 評価指標と算出方法

切替前後の姿勢を比較する基準時刻には，図 3 下部に表記した Gateway の転送先切替 API を呼び出す直前に保持されていた最新の odometry の時刻（切替要求直前のタイムスタンプ）を用い，以降 H と称する．これは切替要求直前の基準であり，Gateway 内部の切替完了の厳密な時刻とは限らない．H 以前の切替元の最後の有効 TF を基準姿勢とし，切替後最初の有効 TF，および H から 5，10，30 秒後に最も近い切替先の有効 TF を比較に用いる．各 TF サンプルでは座標変換を同一時刻に揃えた．

基準サンプルにおける切替元の map 座標系上の姿勢を X_-，odom 座標系上の姿勢を O_-とする．また，切替先の評価時刻tにおける odom 姿勢を$O(t)$，map 姿勢を$X_{\mathrm{dst}}(t)$とする．切替前後の姿勢を比較する際，ロボット自身の移動による姿勢変化を除く必要がある．ROS の odom 座標系は短期間におけるロボットの連続的な移動を表すため，本評価では切替元の基準姿勢から odometry に従って移動した場合の map 姿勢を予測し，切替先の SLAM が出力する姿勢と比較する．まず，基準サンプルから時刻tまでの相対移動量を式 8 で算出する．

$$\Delta O(t) = O_-^{-1}\, O(t) \qquad (8)$$

式 8 の相対移動量を切替元の map 姿勢に合成し，時刻tの予測姿勢を式 9 により求める．

$$\hat{X}(t) = X_-\, \Delta O(t) \qquad (9)$$

この予測姿勢と切替先 Cartographer が出力した姿勢$X_{\mathrm{dst}}(t)$の差を，式 10 の剛体変換$E(t)$として求める．

$$E(t) = \hat{X}(t)^{-1} X_{\mathrm{dst}}(t) \qquad (10)$$

$E(t)$の平面並進成分を$e_x(t)$，$e_y(t)$，yaw 成分を$\psi_E(t)$とし，位置残差および yaw 残差をそれぞれ式 11，式 12 で定

ⓒ2026 Information Processing Society of Japan

義する．yaw 差は−180°から 180°に正規化した最小回転角の絶対値とする．

$$e_p(t) = \sqrt{e_x(t)^2 + e_y(t)^2} \qquad (11)$$

$$e_\psi(t) = |\psi_E(t)| \qquad (12)$$

本評価で算出する残差は，切替元の姿勢から odometry によって予測した姿勢と，切替先の SLAM が出力した姿勢との差を表す．したがって，ロボットの物理的な位置の真値に対する誤差ではない．また，切替後の残差には Cartographer による姿勢推定の更新も反映されるため，切替処理のみに起因する姿勢変化を表すものではない．本評価では，切替基準時刻 H 以降に切替先から取得された最初の有効 TF を用いた残差を「切替直後の残差」とする．

引継ぎ処理時間について，ホスト側の引継ぎ要求から Gateway の転送先切替 API が完了するまでの経過時間を用いる．また，状態の保存（Export），状態の読込（Import）については，各制御処理の開始から完了までの所要時間を用いる．

4.5 結果

以下では各試行の残差を算出し，条件ごとに中央値を示す．図の横軸は評価時点を等間隔に配置している．

直進条件の各時点における条件 A と B の残差中央値を表 2 に，位置残差と yaw 残差の推移をそれぞれ 図 5，図 6 に示す．

表 2 直進条件の残差中央値（A と B）

計測時点	A 位置（cm）	B 位置（cm）	A yaw（°）	B yaw（°）
直後	3.411	1.107	0.215	0.434
5 秒後	3.585	1.007	0.777	0.125
10 秒後	4.721	0.731	1.010	0.356
30 秒後	4.886	5.129	0.136	0.250

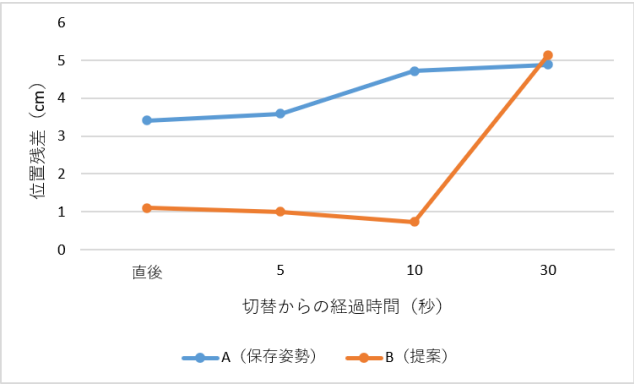

図 5 直進条件の位置残差中央値の推移

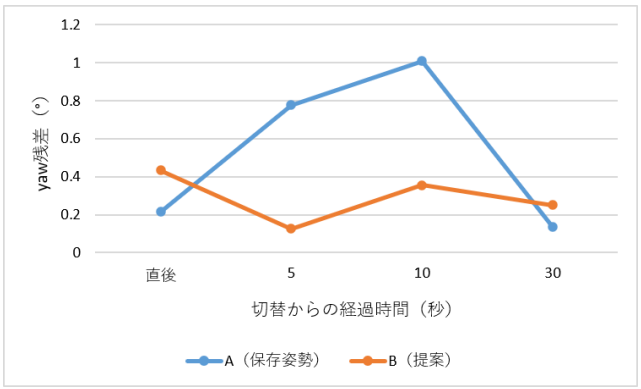

図 6 直進条件の yaw 残差中央値の推移

直進条件の切替直後の位置残差中央値は，A の 3.411 cm に対し B では 1.107 cm となり，約 67.6%減少した．また，位置残差について，条件 A の最小値は 2.303 cm，条件 B の最大値は 1.495 cm であり，条件 B の全試行で条件 A のいずれの試行よりも小さい残差が得られた．B の位置残差中央値は 5，10 秒後も A より小さかったが，30 秒後は A が 4.886 cm，B が 5.129 cm となった．一方，切替直後の yaw 残差中央値は A の 0.215°に対し B が 0.434°と大きく，この時点の改善は確認されなかった．

次に，旋回条件の各時点における条件 A と B の残差中央値を表 3 に，位置残差と yaw 残差の推移をそれぞれ図 7，図 8 に示す．

表 3 旋回条件の残差中央値（A と B）

計測時点	A 位置（cm）	B 位置（cm）	A yaw（°）	B yaw（°）
直後	2.656	1.421	9.434	3.506
5	9.023	5.324	9.958	4.03
10	1.217	1.463	2.178	2.313
30	7.353	7.847	2.844	2.979

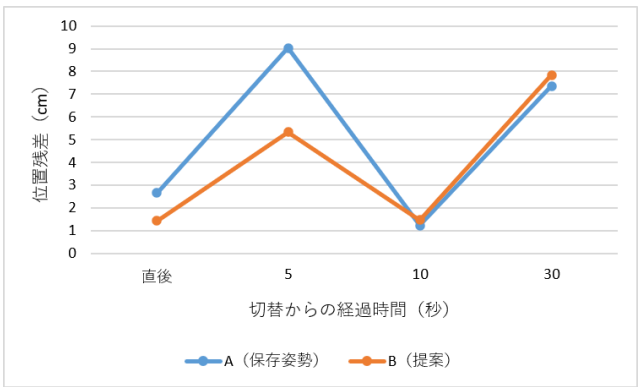

図 7 旋回条件の位置残差中央値の推移

ⓒ2026 Information Processing Society of Japan

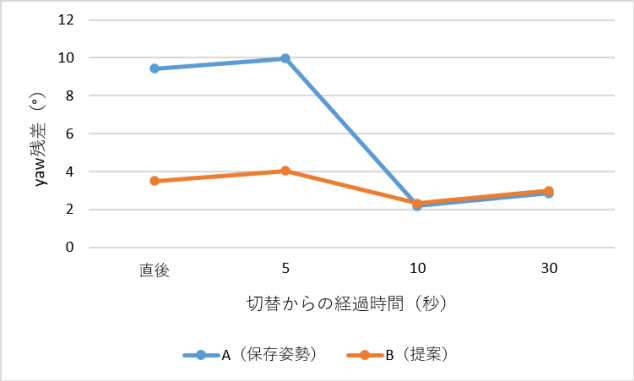

図 8 旋回条件の yaw 残差中央値の推移

旋回条件の切替直後の位置残差中央値は A の 2.656 cm から B の 1.421 cm へ約 46.5%減少し，yaw 残差中央値は 9.434°から 3.506°へ約 62.8%減少した．いずれの指標でも B の全 10 試行が A の全 10 試行を下回った．5 秒後も B の残差中央値は両指標で小さかったが，10 秒後と 30 秒後には A と B が近い値となり，B の一貫した優位性は確認されなかった．

次に，SLAM 状態の引継ぎに要する時間を確認した．Export，Import の各制御時間，および引継ぎ要求開始から Gateway の切替完了までの時間について，条件 A と B の中央値を表 4 に示す．

表 4 引継ぎ処理時間の中央値（ms）

計測項目	直進 A	直進 B	旋回 A	旋回 B
Export 制御	206.4	203.3	255.7	251.6
Import 制御	121.2	122.5	120.9	118.0
引継ぎ要求～切替完了	382.0	372.2	440.6	434.1

引継ぎ要求開始から Gateway の切替完了までの所要時間の中央値は，条件 A と B でそれぞれ，直進時に 382.0 ms と 372.2 ms，旋回時に 440.6 ms と 434.1 ms であった．条件 B では，Export と Import の制御時間が直進と旋回で全体の約 85~88%を占めていた．そのうち，Export 時の SLAM 状態のシリアライズと，Import 時の LoadState による状態復元処理時間を分析したところ，これら 2 つの処理時間の合計が引継ぎ処理時間全体に占める割合の中央値は，直進時に約 81.0%，旋回時に約 80.8%であった．このことから，今回の実装では，SLAM 内部状態のシリアライズと LoadState が引継ぎ処理の主要なコストであることが確認された．一方，A と B の引継ぎ時間には大きな差は見られなかった．

4.6 考察

引継ぎ処理時間の評価から，今回の実装では SLAM 状態のシリアライズと復元が主要な処理コストとなっていることが分かった．また，条件 A と B の引継ぎ処理時間は概ね同程度であり，初期姿勢の設定方法の違いによる大きな時間差は見られなかった．この結果は，今回の連続性改善が引継ぎ処理の高速化によるものではなく，保存後のロボットの移動量を初期姿勢に反映したことによる効果と整合する．ただし，両条件では共通の制御経路を使用しているため，姿勢補正計算単体の追加処理時間を評価したものではない．

切替前と後の姿勢差を初期姿勢に反映する提案条件 B では，条件 A と比較して，切替直後の位置残差中央値が直進時に約 67.6%，旋回時に約 46.5%減少し，旋回時の yaw 残差中央値も約 62.8%減少した．直進条件では位置残差の減少量が 2.304 cm と旋回条件の 1.235 cm より大きく，旋回条件では yaw 残差が 5.928°減少した一方，直進条件では yaw 残差の改善は確認されなかった．これらの結果から，直進条件では位置残差の減少量が旋回条件より大きく，旋回条件では yaw 残差の減少が顕著であった．この傾向は，直進時には並進移動が主体となり，旋回時には方位変化が大きくなるという走行特性と整合している．ただし，走行条件による補正効果の違いを一般化するには，移動速度や旋回角速度を変化させた評価が必要である．また，切替後 10~30 秒では B の優位性が確認されない場合があった．切替後の自己位置推定には初期姿勢だけでなく，新たなセンサデータを用いた Scan Matching や姿勢最適化も影響するため，初期補正姿勢による改善効果が相対的に小さくなった可能性がある．

本提案手法は，切替後の自己位置推定精度を継続的に向上させることではなく，SLAM 状態の引継ぎ時に発生する自己位置出力の不連続を抑制することを目的としている．今回の評価では，既定の引継ぎ条件において，切替直後の位置連続性の改善が確認された．一方，状態保存から切替までの時間差がさらに大きくなった場合の補正効果については，本評価の結果からは明らかではない．

5. まとめ・今後の展望

本稿では，移動中のロボットに対するコンテナ間 SLAM 引継ぎ時の自己位置出力の不連続を抑制するため，保存時の map 姿勢と odom 姿勢，および Gateway が保持する最新の odom 姿勢から補正初期姿勢を求める手法を提案した．単一ホスト上の評価では，保存時姿勢のみを使用する条件 A と比較して，提案手法 B の切替直後の位置残差中央値は直進で 3.411 cm から 1.107 cm へ約 67.6%，旋回で 2.656 cm から 1.421 cm へ約 46.5%減少した．旋回では yaw 残差中央値も 9.434°から 3.506°へ約 62.8%減少した一方，直進の切替直後では yaw 残差の改善は確認されなかった．また，切替後 30 秒まで B が常に優位となる結果ではなかった．これらから，提案手法が今回の走行条件における切替直後の位置連続性向上に寄与することを確認した．

ⓒ2026 Information Processing Society of Japan

今後は，今回の引継ぎ処理時間の分析で判明した，シリアライズおよび LoadState が主要な処理コストとなっていた結果を踏まえ，SLAM 内部状態の保存復元処理の効率化について検討する．

参考文献

[1] 株式会社 マキテック：ロボットシステム（オンライン），入手先〈https://www.makitech.co.jp/conveyor/13/agv.html〉（参照 2026-10-22）．
[2] 大成建設株式会社：自律走行清掃ロボットシステム「T-CleanX」を開発（オンライン），入手先〈https://www.taisei.co.jp/about_us/wn/2023/230807_9616.html〉（参照 2026-10-22）．
[3] H. Durrant-Whyte and T. Bailey.: Simultaneous localization and mapping: part I, *IEEE Robotics & Automation Magazine*, Vol. 13, No. 2, pp. 99-110(2006).
[4] P. Huang, L. Zeng, X. Chen, K. Luo, Z. Zhou and S. Yu.: Edge Robotics: Edge-Computing-Accelerated Multirobot Simultaneous Localization and Mapping, *IEEE Internet of Things Journal*, Vol. 9, No. 15, pp. 14087-14102(2022).
[5] V. K. Sarker, J. Peña Queralta, T. N. Gia, H. Tenhunen and T. Westerlund.: Offloading SLAM for Indoor Mobile Robots with Edge-Fog-Cloud Computing, 2*019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pp. 1-6(2019).
[6] P. Zhang, H. Wang, B. Ding and S. Shang.: Cloud-Based Framework for Scalable and Real-Time Multi-Robot SLAM, *2018 IEEE International Conference on Web Services (ICWS)*, pp. 147-154(2018).
[7] P. Huang, L. Zeng, K. Luo, J. Guo, Z. Zhou and X. Chen.: ColaSLAM: Real-Time Multi-Robot Collaborative Laser SLAM via Edge Computing, *2021 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 242-247(2021).
[8] K. Kubota, Y. Hasuhara and M. Sugaya.: A Scalable Offloading Method for Simultaneous Localization and Mapping Processing in Multi-Robot Environments, *2026 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*, pp. 1-9(2026).
[9] J. Hofer, P. Sossalla, J. Rischke, C. L. Vielhaus, M. Reisslein and F. H. P. Fitzek.: Circular Frame Buffer to Enhance Map Synchronization in Edge Assisted SLAM, *ICC 2023 - IEEE International Conference on Communications*, pp. 210-215(2023).
[10] W. Hess, D. Kohler, H. Rapp and D. Andor.: Real-time loop closure in 2D LIDAR SLAM, *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271-1278(2016).
[11] Redis：Introduction to Redis（オンライン），入手先〈https://redis.io/about/〉（参照 2026-10-22）．
[12] Blanco-Claraco, José Luis.: A tutorial on SE (3) transformation parameterizations and on-manifold optimization, *arXiv preprint arXiv:2103.15980* (2021).
[13] T. Foote.: tf: The transform library, *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, pp. 1-6(2013).
[14] W. Meeussen：Coordinate Frames for Mobile Platforms（オンライン），入手先〈https://reps.openrobotics.org/rep-0105/〉（参照 2026-10-22）．
[15] The Cartographer Authors：StartTrajectory.srv（オンライン），入手先〈https://github.com/ros2/cartographer_ros/blob/ros2/cartographer_ros_msgs/srv/StartTrajectory.srv〉（参照 2026-10-22）．

ⓒ2026 Information Processing Society of Japan