論　文

RT ミドルウェアによる移動ロボットのナビゲーションフレームワークの構築

佐々木　智典[*1)]　中庄　貴之[*2)]　塩沢　恵子[*2)]　坂下　和広[*1)]
村上　真之[*1)]　益田　俊樹[*1)]　森田　裕介[*1)]　小林　祐介[*1)]

Building a mobile robot navigation framework with RT-middleware

Akinori Sasaki[*1)], Takayuki Nakasho[*2)], Keiko Shiozawa[*2)], Kazuhiro Sakashita[*1)],
Masayuki Murakami[*1)], Toshiki Masuda[*1)], Yusuke Morita[*1)], Yusuke Kobayashi[*1)]

This paper describes the development of a navigation framework for mobile robots. The navigation framework is designed for a mobile robot base developed in TIRI. The framework is based on RT-middleware, which allows developers to make software components interoperable among various computing platforms. Such interoperability makes it easy to perform prototyping related to service robot applications.

キーワード： 移動ロボット，ナビゲーション，RT ミドルウェア
Keywords : Mobile robots, Navigation, RT-middleware

1. はじめに

本稿では移動ロボットのナビゲーションフレームワークの構築について述べる。

いわゆるサービスロボットあるいは生活支援ロボット等に分類されるロボットには，家電製品の延長にある掃除ロボットのように，市場を確立しつつある製品もある。しかし，製造工程の自動化に利用されている産業用ロボットに比べて，広く普及しているとは言い難い段階にある。DARPA Urban Challenge 等のロボティクス関連研究[(1)]から派生した自動運転車（self-driving car）に見られるように，技術的に高度な自動化・自律化が達成可能となってきたが，一方，実際の事業として展開するに当たっては，開発・製造・運用コストと収益性の兼ね合いを含めて，妥当なレベルのサービス，製品を達成することは困難である。したがって，サービスロボットの開発にあたっては，ハードウェア，ソフトウェア，サービス運用等の様々なレベルで試行錯誤の過程を経ざるをえない。

このようなプロトタイピングの段階においては，開発の容易性が重要である。一方で，高度な自動化・自律性を達成しようとすると，ハードウェア，ソフトウェアの規模も大きくなる。これを整理・整頓し，かつ多様な組み合わせでのプロトタイピングを実現するには，様々なレベルでのモジュール化やモジュール間の相互作用の仕組みが必要となる。

事業名　平成 26 年度　共同研究
*1) ロボット開発セクター
*2) 株式会社 アドイン研究所

このようなモジュール化を実現するソフトウェア基盤として RT ミドルウェア[(2)]や ROS[(3)]が提案されており，様々なシステム構築が行われている。RT ミドルウェアや ROS の実装はオープンソースソフトウェアとして公開されており，これを基盤とするモジュールにも，オープンソースソフトウェアとして公開されているものがある。RT ミドルウェア，ROS のいずれもソフトウェアのモジュール化とモジュール間の連携を実現する仕組みとして類似した部分もあるが，その仕様や関連ツールなどを含めて様々な違いがある。本研究では，既存のモジュールを連携させることを考慮して，RT ミドルウェアの標準的実装である OpenRTM-aist を利用した。

本稿は以下のように構成される。第 2 章では，本稿で想定するナビゲーション（自律移動機能）の概説を示す。次に，第 3 章において開発したフレームワークにおけるハードウェアとソフトウェアの構成について述べ，第 4 章でその使用例を示す。最後に第 5 章において結論を述べる。

2. ロボットの自律移動の基本

移動ロボットを活用したサービス（例えば，道案内）を実現しようとする場合に，ロボットに自律移動の機能，すなわちナビゲーションが求められる[(5), (6)]。

ロボットの自律的な移動を次のような段階に分けて考える。

(1) 地図構築（mapping）
(2) 測位（localization）
(3) 経路計画（path planning）
(4) 経路追従制御（path tracing）

(1) 地図構築は，ロボットが移動する空間にある壁や柵などの静的な物体の配置のデータを準備する作業である。地図は手動で作成するか，あるいはロボットに搭載したセンサのデータから半自動的に作成する。

(2) 測位は，ロボットが環境（地図）の中でどこにいるかを自ら測定する処理である。屋外であれば GPS のような外部装置を利用する測位方法が使用可能であるが，ここではそのような装置を使わずに，ロボットに搭載したセンサのみで自己位置推定を行うものとする。自己位置推定は，センサデータや推定時点までの動作の系列を基に行う。

(3) 経路計画は現在位置から目標位置に至るまでの経路を決定する処理である。経路を決定する基準としては，壁などに衝突しないこと，経路全体の長さを短くすること（遠回りをしないこと）が挙げられる。このためには地図の情報が必要である。

(4) 経路追従制御は，計画した経路をたどるように機体運動の制御を行う過程である。これには時時刻刻と変化するロボットの位置を測定しつつ，適宜，所定の経路をたどるように運動の方向を調節する。

2. 1 地図構築　地図は，ロボットが移動する空間にある壁や柵などの静的な物体の配置のデータである。地図の具体的表現として，占有度格子地図（occupancy grid map）を利用する。占有度格子地図は空間の離散的な表現であり，2 次元配列 $m[x, y]$ で表現される。ここで x, y は格子の座標とする。各格子につき，$0 \leqq m[x, y] \leqq 1$ の範囲の値を割り当てる。この値を画素値に対応付けると，地図 $m[x, y]$ はグレースケール画像として表現される。

占有度格子地図 $m[x, y]$を構築する方法としては，画像編集ソフトウェアなどを利用して，手動で作成する方法と，ロボットの搭載センサのデータから半自動的に構築する方法がある。後者は SLAM（Simultaneous Localization And Mapping）と総称される[5), (6)]。

2. 2 測位　ロボットの運動を制御するには，その位置を測定する必要がある。自己位置推定は，ロボットに搭載したセンサデータを基に，地図座標系におけるロボットの位置を推定する問題である。本フレームワークでは確率統計に基づく自己位置推定手法である MCL （Monte-Carlo Localization）を使用する。

センサとして次の二つを利用する。一つはロボット機体の車輪回転角度を検出するロータリエンコーダである。もう一つは，赤外レーザにより周囲の物体への距離を測定するレーザレンジファインダ（laser range finder, LRF）である。例えば，LRF の一種である URG-04LX-UG01（北陽電機（株））は，100 msec に 1 回のスキャンごとに，240 deg の範囲の 682 点分の距離を測定する。

確率統計に基づく自己位置推定は，動作データ u_t および観測データ z_t の系列（$u_{1:t} \equiv \{ u_1, \ldots, u_t \}$, $z_{1:t} \equiv \{ z_1, \ldots, z_t \}$），そして地図 m が与えられたときの現在位置（および姿勢）x_t に関する事後確率分布 $p(x_t \mid z_{1:t}, u_{1:t}, m)$ を推定する問題として定式化される。動作データ u_t は，ロータリエンコーダのデータから，機体の機構に基づいて計算される。また，観測データ z_t として LRF のデータが使用される。

MCL においては事後確率分布 $p(x_t \mid z_{1:t}, u_{1:t}, m)$ を，標本（パーティクル）の集合によって表現する。MCL では時間の進行とともに，取得される動作データ u_t，観測データ z_t を用いて，ベイズ則に基づく反復計算により，パーティクルの集合を更新する。その結果のパーティクルの集合が 事後確率分布 $p(x_t \mid z_{1:t}, u_{1:t}, m)$ に対応しており，その期待値がロボットの推定位置として計算される。

2. 3 経路計画　次にロボットの推定位置に基づいて，目標位置に至るまでの経路を計画する。この経路計画問題は，マルコフ決定過程として定式化される。この下で，障害物に衝突せず，かつ，目標地点までの距離がなるべく短くなるような経路について高い値を与える評価関数を用いる。これを最大化することにより経路を決定する問題を解く。単純化のため，距離として格子距離を採用するため，生成される経路は必ずしも直感的な最短経路とはならないが，経路追従制御において機構制約から運動学的に無理がないような運動を行うように制御することで対処する。

2. 4 経路追従制御　次に計画した経路をたどるように機体運動の制御を行う（障害物がある場合には一時停止または回避を行う）。目標地点に到達することのみを目的とすると，所定の経路追従制御は必要とは限らない。例えば，反射運動的な方式として，障害物の検出と，瞬時・局地的な移動先決定を繰り返す方法がありうる（ただし，なるべく最終目標地点に近づくようなバイアスをつけて移動先を選択する）。しかしながら，道案内をさせるような応用においては所定の経路をたどることが求められる。

制御の目標値は，計画した経路，すなわち経由点の系列から決定する。一方，フィードバック量であるロボットの位置は自己位置推定によって与えられる。

3. ハードウェアおよびソフトウェアの構成

本章では開発したフレームワークのハードウェア及びソフトウェアの構成を示す。

3. 1 ハードウェア構成　制御対象とするロボットのハードウェアは以下により構成される。

(a) T 型ロボットベース: 都産技研において開発
(b) LRF: 北陽電機（株）SCIP2.0 規格に準拠の製品
(c) 上位コントローラ: 例えばノート PC（OS は Ubuntu 14.04 LTS）
(d) USB 接続ゲームパッド（HID 規格準拠製品）

T 型ロボットベース以外はすべて市販品である。ゲームパッドは地図構築の際など，運用作業者による手動操作が必要なときに利用する。

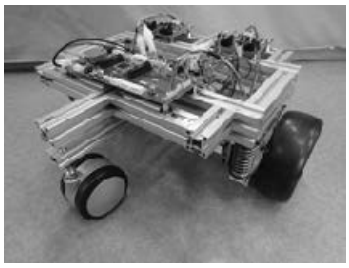

図 1. T 型ロボットベース

3. 2　移動ロボット　本フレームワークで制御対象とする移動ロボットは都産技研において開発した T 型ロボットベースである（図 1）。この移動ロボットは 4 個の車輪を持ち，このうち二つが駆動輪，残り二つが受動輪である。二つの駆動輪は同一の軸まわりに回転するように配置され，それぞれ個別のモータにより駆動される。この機構では，二つの駆動輪を同速度で逆方向に駆動することでその場での旋回運動が可能である。

ロボットベースには二種類の基板が搭載されている。一つは制御基板であり，もうひとつはモータドライバ基板である。制御基板は PSoC（Programmable System-on-Chip, Cypress Semiconductor Corp.）を備え，ファームウェアを実行する。制御基板は二つのモータの回転速度の制御を行い，また，モータの回転方向および速度を検出するロータリエンコーダの信号処理も行う。

上位コントローラは，制御基板と USB 2.0 インタフェースにより接続される。上位コントローラは制御基板（PSoC）に左右の車輪の回転速度制御の目標値を送信する。一方，制御基板に接続された各種センサのデータを受信する。センサデータにはロータリエンコーダのデータが含まれる。これは各駆動輪の回転角度に対応し，これを基にロボットの運動の推定，すなわちオドメトリの計算が可能である。

3. 3　ソフトウェア構成　制御対象とするロボットのソフトウェアは以下により構成される。

(a) ロボットベースと通信するコンポーネント
(b) 自律移動を管理するコンポーネント
(c) LRF のデータを取得するコンポーネント
(d) ゲームパッドの操作を取得するコンポーネント
(e) タスクを管理するコンポーネント（応用ごとに異なる）
(f) 地図構築ツール（オフライン実行）

これらの実行時の関係を図 2 に示す。Intel Core i7 のノート PC でこれらを実行した際の CPU 占有率は，総計で最大でも 50~60%程度であった。

これらのコンポーネントは C++言語により実装しているが，OpenRTM-aist が提供する通信基盤によって，よりプロトタイピングに適した Python 言語で実装されたコンポーネントとも連携が可能である。

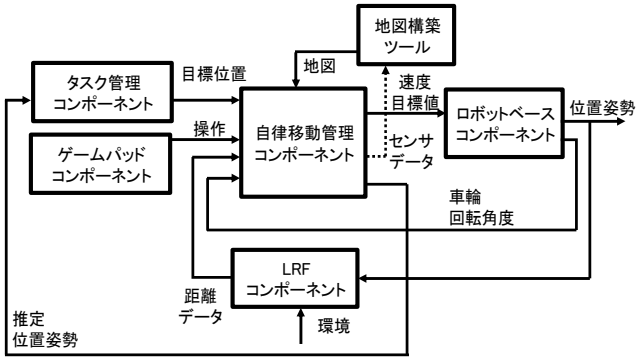

図 2. フレームワークのソフトウェアの構成

フレームワークの実装においては，既存のライブラリである，MRPT（Mobile Robot Programming Toolkit）[7] を利用した。設計にあたり，MRPT の他には Karto SDK[8], Player[9], ROS navigation stack[10] など既存のライブラリの利用を検討した。ツールの充実性，導入容易性（プロプライエタリ/オープンソース/ライセンスの観点）などから MRPT を選定した。

4.　フレームワークの適用実験

本章ではフレームワークを使用してナビゲーションを実施した事例とその結果を示す。

4. 1　地図構築　最初に，ロボットを運用する環境の地図の構築を行った。同一の環境でナビゲーションを実施するのであれば，この作業は事前に一度だけ実行すればよい。まず，ロボットを運用環境において走行させ，センサデータ（オドメトリおよび LRF のデータ）を時系列で取得した。

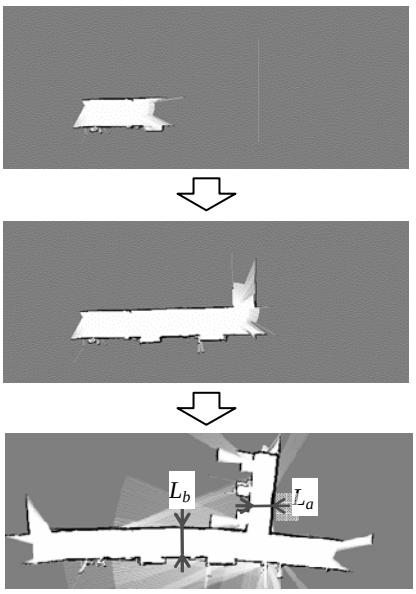

図 3. SLAM による地図構築の例

(a) 来訪者向け案内図。太線内が構築した地図に対応　(b) 図 (a) の T 字分岐から左側を見て撮影

図 4. 計測を行った環境

表 1. 地図の一部の寸法

	画素単位での測定	cm への換算	巻尺測定
L_a	28 pixels	196 cm	198 cm
L_b	35 pixels	245 cm	248 cm

次に，このデータを用いて，SLAM により占有度格子地図を構築した。構築の事例を図 3 に，計測を行った環境を図 4 に示す。図 3 において最上段の地図は，ロボットが動き始めて数秒の段階に対応し，データ収集の進展とともに地図が拡張されて構築される。図 4 (b) に示す左側の壁には，ガラスの部分があり，これは LRF では検出されない。図中の L_a, L_b の寸法を，地図上での画素単位での測定と，実環境上での巻尺による測定とにより比較すると表 1 のようになった。なお, この地図構築においては地図の分解能を 7 cm/pixel と設定した。

4. 2　経路計画　次に，構築された地図，現在位置，そして目標位置のデータを基に，ロボットの走行経路を計画した。目的位置は，ユーザからの指示により与えた。一方，現在位置は自己位置推定によって与えられる。目的位置は，ロボットにより実現するサービスに依存するが，運用中に変動するデータであり，経路計画は目的位置の変更に応じて繰り返し行われる。

経路計画の例を図 5 に示す。図中の小矢印の先端は経由点に対応し，この点列をたどるようにロボットを移動させることが制御の目標となる。

4. 3　自己位置推定および経路追従制御　事前に計画された経路に基づいて，ロボットが自律移動を行わせた。図 6 に自己位置推定の様子を示す。この図中において地図の空白部分にプロットされた点群がパーティクルの集合を表し，センサデータの取得と推定の更新処理に応じてその分布は変化する。この分布の期待値に相当する位置・姿勢がロボットの推定位置・姿勢に相当する（パーティクルは姿勢についての成分も持っている）。図中では右上から左下に向かってロボットが移動している。

5. まとめ

本稿では，移動ロボットのナビゲーションフレームワークの構築について述べた。本フレームワークにより，制御対象のロボットベースに搭載されたセンサから収集されるデータに基づきロボットの自律移動が実現される。本フレームワークは OpenRTM-aist をソフトウェアの基盤とし，コンポーネントを単位として構築されており，サービスロボットのプロトタイピングに利用可能である。

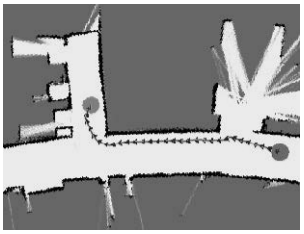

図 5. 経路計画の例

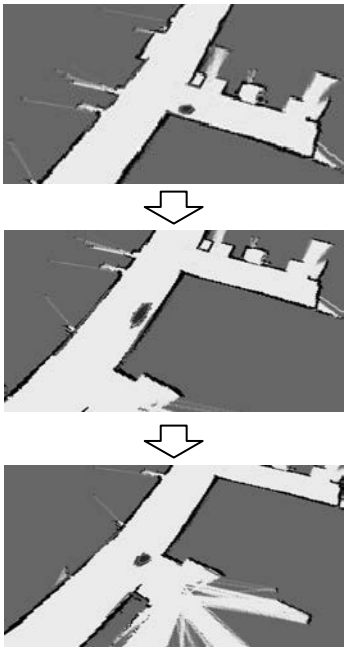

図 6. MCL による自己位置推定の例

（平成 28 年 7 月 4 日受付，平成 28 年 7 月 29 日再受付）

文　献

(1) Anna Petrovskaya and Sebastian Thrun: “Model Based Vehicle Detection and Tracking for Autonomous Urban Driving", Autonomous Robots, Vol.26, No.2-3, pp.123-139 (2009)
(2) 安藤慶昭: 「OMG における Robotic Technology Component（RTC）および関連仕様の標準化動向」, 日本ロボット学会誌, Vol.29, No.4, pp.333-336 (2011)
(3) Steve Cousins and Brian Gerkey: “Milestones: First ROScon and OSRF”, IEEE Robotics & Automation Magazine, Vol.3, No.19, pp. 14-15 (2012)
(4) OpenRTM-aist: http://www.openrtm.org/openrtm/ja/, 2016.07.01閲覧
(5) Sebastian Thrun, Wolfram Burgard, and Dieter Fox, “Probabilistic Robotics”, MIT Press (2005)
(6) 友納正裕:「[解説] 移動ロボットのための確率的な自己位置推定と地図構築」, 日本ロボット学会誌, Vol.29, No.5, pp.423-426 (2011)
(7) MRPT (Mobile Robot Programming Toolkit): http://www.mrpt.org 2016.07.01 閲覧
(8) Karto SDK:　http://www.kartorobotics.com, 2016.07.01閲覧
(9) Player Project: http://playerstage.sourceforge.net, 2016.07.01閲覧
(10) ROS navigation stack. http://wiki.ros.org/navigation, 2016.07.01閲覧