解 説

ROS を用いた自律走行

Autonomous Navigation with ROS

原 祥 尭* *千葉工業大学未来ロボット技術研究センター（fuRo）
Yoshitaka Hara* *Future Robotics Technology Center, Chiba Institute of Technology

1. はじめに

ROS は，ロボット開発に有用なミドルウェアやフレームワークを提供している．開発の基盤となる機能だけでなく，自律走行などの機能も持つ．自律走行は，多種の移動ロボットにとって不可欠な，重要な機能である．自律走行には，自己位置推定，地図生成（SLAM: Simultaneous Localization and Mapping），障害物検出，経路・動作計画などの各技術が必要となる．本稿では，ROS が提供するこれらの自律走行機能について解説するとともに，これを用いた自律移動ロボットの開発事例を紹介する．

2. 自律走行機能を持つ ROS パッケージ

ROS では，様々な機能がパッケージの形式で提供されている．また，特定の目的に必要なパッケージ群を，メタパッケージとしてまとめている．ROS の既存パッケージを用いて自律走行を行う場合，navigation [1] と slam_gmapping [2] の二つのメタパッケージを用いる場合が多い．ほかにも有用なパッケージは存在するが，本稿ではこれらを対象とする．

図 1 に，navigation と slam_gmapping メタパッケージの構成を示す．この二つで，自律走行に必要な機能がひととおり提供される．また ROS では，C++や Python が主に使われるが，これらのパッケージは C++で実装されている．アルゴリズムの詳細は後述する．

navigation メタパッケージは，既存地図での自己位置推定，障害物情報の管理，経路・動作計画の機能を持つ．実行ファイル（ROS ノード）として，map_server，amcl，move_base などが提供される．map_server は，既存地図を配信するノードである．amcl は，Particle Filter による自己位置推定を行うノードである．move_base は，移動タスクを実行するノードである．具体的には，costmap_2d がコストマップとして障害物情報を管理し，nav_core のインタフェースで計画を立てる．nav_core の実装は，動的に変更可能な仕組みになっている．

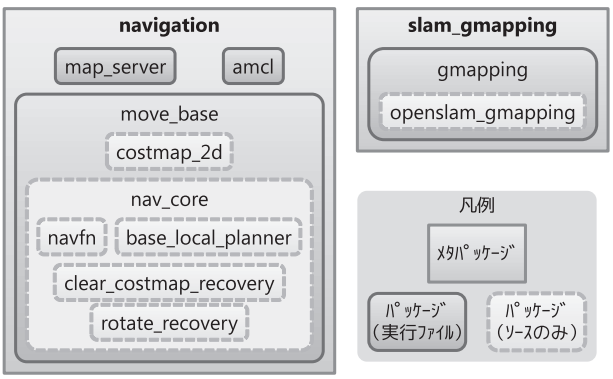

図 1 navigation と slam_gmapping メタパッケージの構成

slam_gmapping メタパッケージは，SLAM による地図生成機能を持つ．実行ファイルとして，gmapping が提供される．gmapping は，OpenSLAM で公開されている Rao-Blackwellized Particle Filter による SLAM [3] の ROS ラッパーである．

自律走行を行うワークフローの例は，次のとおりである．1）移動ロボットを手動走行させてセンサデータを取得，bag ファイルに記録．2）bag ファイルを再生し，slam_gmapping メタパッケージを使用してオフラインで地図生成†．3）生成した地図に基づき，navigation メタパッケージを使用して自律走行．あるいは，slam_gmapping で地図を生成しながら，同時に navigation で自律走行することもできる．

3. 自己位置推定，SLAM のアルゴリズム

amcl と gmapping では，Bayes Filter 系の手法が用いられている．ロボット位置や地図を，ベイズ推定に基づいて確率的に求める．Bayes Filter の実装には，Extended Kalman Filter（EKF）や Histogram Filter（HF）などもあるが，ここではともに Particle Filter（PF）の系列が使われている．

amcl のアルゴリズムは，PF による Monte Carlo Lo-

原稿受付 2017 年 3 月 31 日
キーワード：ROS, Navigation, Localization, Mapping, SLAM, Path Planning, Motion Planning, Tsukuba Challenge
*〒 275–0016 習志野市津田沼 2–17–1
*Narashino–shi, Chiba

†bag ファイルの再生でなく，1）と同時にオンラインで実行してもよい．

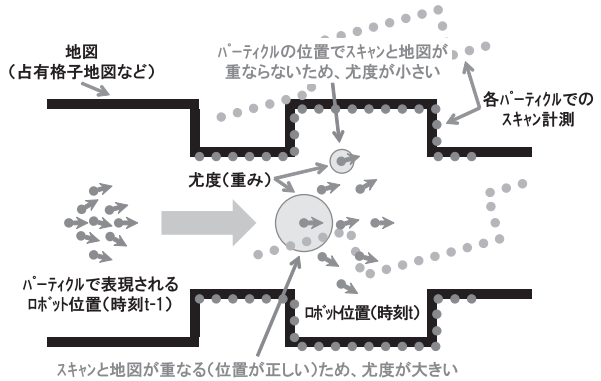

図 2　Particle Filter による Monte Carlo Localization

calization（MCL）[4] である．**図 2** に，MCL の概要を示す．ロボット位置の確率分布を，サンプリングした複数のパーティクルで近似する．処理手順は，次の 3 ステップである．1）時刻 $t-1$ の各パーティクルをオドメトリなどによる動作モデルで移動して，時刻 t の事前確率を予測する．2）レーザスキャンなどによる計測モデルで尤度を計算し，各パーティクルに重み付けする．3）重みに基づいて各パーティクルをリサンプリングする．スキャンと地図が重なるパーティクルは尤度が大きくなり，多くの複製を残す．一方で，尤度が小さいパーティクルは消滅する．これにより，ロボット位置を修正（更新）して時刻 t の事後確率を求める．

amcl には，地図形式として占有格子地図，動作モデルとしてオドメトリ動作モデル，計測モデルとしてビーム計測モデルと尤度場計測モデル [5] が実装されている．さらに，推定位置の不確かさに応じて，パーティクル数を適応的に調整する仕組みを持つ．KLD サンプリングを用いた Adaptive MCL [6] という手法で，amcl の名前の由来にもなっている．また，真の位置のパーティクルを喪失する問題に対処するため，一部のパーティクルをランダムに配置する Augmented MCL [7] も実装されている．

gmapping のアルゴリズムは，Rao-Blackwellized Particle Filter（RBPF）[8] による Grid-Based SLAM [9] である．RBPF は PF の一種であり，SLAM においてはロボット位置を PF で，地図をほかの何らかのフィルタで推定するものだと捉えればよい．ここでは，各パーティクルが現在のロボット位置だけでなく過去の走行軌跡も保持できる性質を利用している．すなわち，パーティクルごとの走行軌跡に基づいて，各パーティクルで独立に地図を推定する．パーティクルの数だけ，地図の仮説が生成される．ランドマーク特徴地図を推定する場合は Extended Kalman Filter が，占有格子地図を推定する場合は Binary Bayes Filter が用いられる．gmapping は後者に該当し，確率的な占有格子地図を求める Grid-Based SLAM である．

†FastSLAM 1.0 [11] には，この工夫は導入されていない．

表 1　自己位置推定と SLAM 手法の分類

	スキャンマッチング系		Bayes Filter 系	Graph-Based SLAM 系
性質	逐次最適化、オンライン		フィルタリング、オンライン	全体最適化、オフライン
手法の概要	点群を最適化計算で位置合わせ		事前確率と尤度を確率的に融合	ﾛﾎﾞｯﾄ位置やﾗﾝﾄﾞﾏｰｸ位置（地図）を表すｸﾞﾗﾌを最適化
	詳細ﾏｯﾁﾝｸﾞ	大域ﾏｯﾁﾝｸﾞ		
	初期位置あり、ﾕｰｸﾘｯﾄﾞ空間で対応付け	初期位置なし、特徴量空間で対応付け		
手法の例	ICP、NDT など	Spin Image、FPFH、PPF、SHOT など	EKF、HF、**PF**、**RBPF** など	ﾎﾟｰｽﾞ調整、ﾊﾞﾝﾄﾞﾙ調整、完全 SLAM

RBPF を用いた gmapping は，PF による Monte Carlo Localization と類似する部分も多いが，主に以下の点が異なる．まず根本的に，gmapping は SLAM 手法であり，地図を生成しながら自己位置推定を行う．特に，パーティクルごとに地図を推定し，計測モデルで尤度を計算する際は各パーティクルが持つ地図を利用する．これは，RBPF による SLAM の一般的な性質である．一方で MCL は，既存の一つの地図での自己位置推定を行う．また gmapping の計測モデルは，単に尤度を計算するのではなく，パーティクルごとにスキャンマッチングを行う [9]．これは，サンプリングする提案分布として動作モデルだけでなく計測モデルも利用することで，必要なパーティクル数を削減する FastSLAM 2.0 [10] の工夫† を，Grid-Based SLAM に適用したものである [12]．amcl には実装されていないが，同様の工夫を自己位置推定に適用した Mixture MCL [13] という手法も存在する．

表 1 に，自己位置推定と SLAM の手法を種類ごとに整理した [14]．amcl の PF による Monte Carlo Localization や，gmapping の RBPF による Grid-Based SLAM は，数ある手法のうちの一部に過ぎない．これらの手法が，どのような性質を持つかを意識することは重要である．特に gmapping に関しては，センサ視野が不足する場合などに，そのままでは性能が不十分なことも多い [15]．

4.　経路・動作計画のアルゴリズム

move_base のアルゴリズムは，基本的に Global Dynamic Window Approach（Global DWA）[16] と呼ばれる枠組みである．まず大域的な経路計画でグローバルパスを求め，それに基づいて局所的な動作計画でローカルパス（走行速度）を求める，2 段階の手法である．それぞれの段階で，障害物を回避してゴールに到達する計画を立てる．この経路・動作計画は，占有格子地図の上で行われる．

まず costmap_2d パッケージが，レーザスキャンなどを用いて二次元／三次元の占有格子地図を生成する．ここでの占有格子地図は，Grid-Based SLAM とは異なり，確率的ではない．占有／自由／未知の 3 状態で障害物情報を管理する．また計画の前処理として，占有格子地図に基づい

て二次元コストマップを生成する．コストマップは格子地図の一種で，各セルが障害物からの距離に基づくコストを持つ．障害物に近いセルはコストが高い．また同時に，占有セルをロボットの大きさで膨張させた，(x, y) の 2 自由度コンフィグレーション空間になっている．これにより，コストマップ上でロボットを 1 点として扱える．大域的な経路計画に用いるグローバルコストマップと，局所的な動作計画に用いるローカルコストマップの 2 種類をつくる．

続いて nav_core パッケージでは，経路・動作計画のインタフェース（C++の純粋仮想関数）のみが定義されている．実装には複数のパッケージがあり，ROS の pluginlib という機能で実行時に切り替えることができる．

大域的な経路計画の実装は，navfn，global_planner，carrot_planner の三つのパッケージがある．navfn は，ダイクストラ法で経路計画を行う．global_planner は，A*またはダイクストラ法で経路計画を行う．carrot_planner は，ゴールまで単純に直進する経路計画を行う．大域的な経路計画は，グローバルコストマップ上で行われる．navfn と global_planner では，Navigation Function [17] と呼ばれるポテンシャル場に基づき，グラフ探索で経路を計画する．ゴールまでの距離と，障害物からの距離に基づくコストの両方を考慮することで，できるだけ最短な経路で，かつ障害物から離れたグローバルパスを求める．この段階ではロボットの方向を考慮せず，円形と仮定した経路が計画される．よってグローバルパスは，方向によっては障害物と衝突する可能性も残る．

局所的な動作計画の実装は，base_local_planner，dwa_local_planner の二つのパッケージがある．ともに，基本は Dynamic Window Approach（DWA）[18] による動作計画を行うが，実装の詳細が異なる．**図 3** に，DWA の概要を示す．局所的な動作計画は，ローカルコストマップ上で行われる．DWA は，ロボットのキネマティクス（運動モデル）を考慮した軌跡で，かつ現在速度からダイナミクス（加速度）を考慮して実行可能な複数のローカルパス候補を，速度の空間でサンプリングする．これらの候補から評価関数により，グローバルパスに追従しつつ，障害物を回避するローカルパス（走行速度）を求める．ここでは，ロボットの方向も考慮した評価により，動作が計画される．

また，ロボットが動けないスタック状態からの復帰動作も，nav_core のインタフェースで定義されている．実装には，clear_costmap_recovery，rotate_recovery，move_slow_and_clear の三つのパッケージがある．clear_costmap_recovery は，ロボット周辺のコストマップをクリアすることで，動けない状態から復帰を試みる．rotate_recovery は，ロボットが旋回して障害物を測定し直すことで，復帰を試みる．move_slow_and_clear は，コストマップをクリアした後に低速で移動して復帰を試みる．

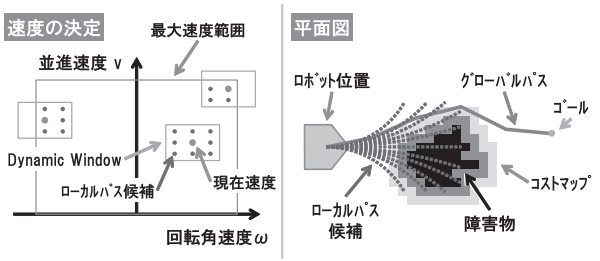

図 3 Dynamic Window Approach による動作計画

5. navigation，slam_gmapping にない機能

navigation と slam_gmapping メタパッケージは様々な機能を提供するが，できないことも多数ある．特に自己位置推定と SLAM に関しては，表 1 に示したように，ほかにも多くの手法が存在する．これらのパッケージにない機能の例としては，以下が挙げられる．

- スキャンマッチング（単体）
- Graph-Based SLAM
- PF による三次元空間 6 自由度での自己位置推定（EKF は robot_pose_ekf パッケージで可能）
- SLAM による三次元空間の地図生成
- 段差などの三次元障害物検出
- 移動障害物の検出と追跡
- (x, y, θ) の 3 自由度コンフィグレーション空間
- 三次元空間 6 自由度での経路・動作計画

このような機能を利用したい場合，自分で新規に実装するか，ほかの既存パッケージを探して使用する必要がある．

SLAM に関しては，RBPF よりも Graph-Based SLAM のほうが性能がよく，現在の主流になっている．Graph-Based SLAM のシステム構築には，フロントエンドとバックエンドと呼ばれる複数の要素の統合が必要である [19]．最近の Graph-Based SLAM のパッケージとして，例えば Google による cartographer [20] がある．なお国内でも，高性能な SLAM システムが提案されている [21]．

6. つくばチャレンジでの自律移動ロボットの開発

つくばチャレンジ [22] は，屋外歩道環境での 1 [km] 以上の自律走行実験の場であり，一般の人々がいる日常の中で行われる．2007 年から毎年開催され，2013 年以降は探索対象を発見する課題も追加された．

表 2 に，つくばチャレンジでの ROS の利用状況を示す†．筆者は 2009 年以降，何度か課題達成したが，当初は ROS を使用していなかった．2014 年に初めて，ROS を用いたロボットで課題達成した．ここでは，既存パッケージを活用してシステムを構築した．なお同ロボットは，2015 年に

†レポートなどに基づく独自集計であり，誤りを含む可能性がある．

表 2 つくばチャレンジでの ROS の利用状況

	'07	'08	'09	'10	'11	'12	'13	'14	'15	'16
参加台数	33	50	72	70	69	36	47	54	56	62
内，**ROS**	0	0	0	1	4	3	6	13	21	33
課題達成	3	1	5	7	6	5	3	4	0	1
内，**ROS**	0	0	0	0	0	0	0	1	0	1

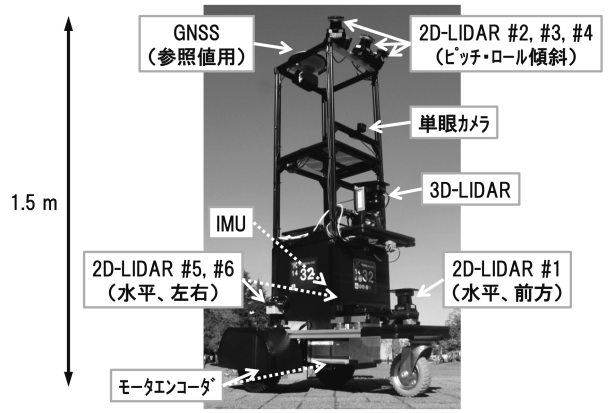

図 4 自律移動ロボット “Rossy”

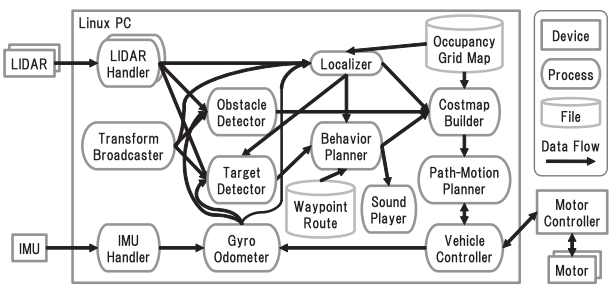

図 5 自律走行システムの構成

雨天で課題達成がゼロの中，探索対象の全発見を唯一達成した．また 2016 年にも，唯一の課題達成を実現している．

図 4 に，ROS で開発した自律移動ロボットの外観を示す．また**図 5** に，自律走行時のシステム構成を示す．navigation と slam_gmapping メタパッケージを使用しているが，改造や新規パッケージの追加を行った．

追加した機能は，主に以下である．まず gmapping を改造することで，屋外の広大な環境でも，測定距離 30 [m] 程度の LIDAR で SLAM を実現した [15]．また，RBPF の各パーティクルが走行軌跡を保持する性質を利用し，教示経路のウェイポイントも自動生成している．**図 6** に，生成した地図とウェイポイントの例を示す．次に，自己位置推定の基礎として，IMU の自動キャリブレーション機能を持つジャイロ併用オドメトリを新規に実装した．屋外の歩道には多様な物体が存在するため，段差などにも対応した三次元障害物検出も開発した．**図 7** に，障害物検出の様子を示す．ほかにも，探索対象の発見や，教示経路を走行しつ

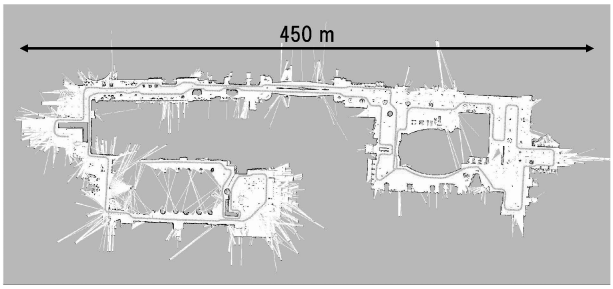

図 6 つくばチャレンジにおいて生成した地図とウェイポイント

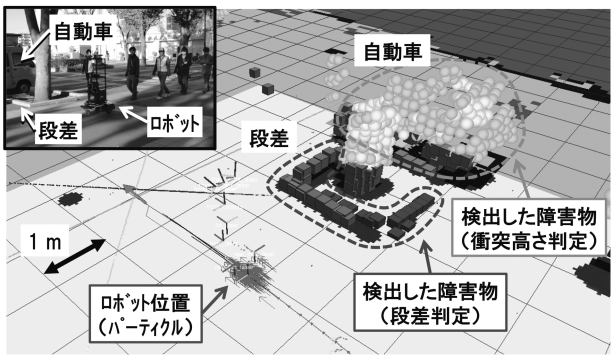

図 7 段差判定と衝突高さ判定による三次元障害物検出

つ探索対称に接近する行動計画も実装した．音声再生による状態通知も実装し，有用だった．最終的に使用しなかったが，人混みに対処するため，高所のランドマークを利用する自己位置推定と SLAM も実装している．

これらの開発により，つくばチャレンジでの課題達成を実現した．しかし，この段階での自律走行の成功率は 8 割程度であり，十分とは言い難い．実用的に使えるレベルにするには，自律走行の各技術を深化させる継続的な研究開発が必要だと考えている．

7. ROS の様々な使い方

一口に「ROS を使う」と言っても，その使い方には様々なレベルがある．ROS が想定している基本的な使い方は，既存パッケージをそのまま起動する方法であろう．ROS はメッセージ型が比較的よく整理されているため，多くの既存パッケージを接続して使うことができる．ただし，アルゴリズムのパラメータ調整には，各手法の中身の理解が必須である．ブラックボックスとして使用しても，望ましい結果が得られるとは限らない．

既存パッケージを C++や Python の API で使用したり，中身を改造して使用する方法もある．アルゴリズムの内部を扱えるため自由度は高い．しかし，ライブラリとして使うには整理が不十分なパッケージも多い．個人的には，自律走行などの上位の既存パッケージは，サンプルプログラムとして捉えるくらいのほうがよいのではと考えている．

新規パッケージを自分で実装する場合にも，標準的なメッ

セージ型に従えば，既存パッケージと連携できる．また，ROS の基盤となるプロセス間通信（IPC），データ記録と再生の枠組み，ビルドツールや可視化ツールなどの開発ツール群は非常に便利であり，これらだけでも有用性は高い．

一方で，ROS を使わないほうがよい状況もある．根本的な部分で機能や性能が不足し，異なる設計が求められる場合には，ROS ではない自前のフレームワークが必要となる．今後は，ROS 2 の発展にも期待したい．

8. おわりに

本稿では，ROS の自律走行機能について解説するとともに，これを用いた自律移動ロボットの開発事例を紹介した．自律走行機能のより具体的な部分は文献 [23] [24] を，また自己位置推定と SLAM の一般的なアルゴリズムの説明は文献 [14] の連載を参照していただければ幸いである．今後，実用的な自律走行システムを構築するためには，要素技術の研究開発の発展とともに，フレームワークを用いて各技術を統合する設計もますます重要になると考えられる．

謝 辞 つくばチャレンジの開発では，筑波大の坪内孝司教授，西田貴亮氏，安藤大和氏らにご協力いただいた．

参考文献

[1] ROS Index—navigation, http://rosindex.github.io/p/navigation/
[2] ROS Index—slam_gmapping, http://rosindex.github.io/p/slam_gmapping/
[3] OpenSLAM — GMapping, http://openslam.org/gmapping.html
[4] F. Dellaert, D. Fox, W. Burgard and S. Thrun: “Monte Carlo Localization for Mobile Robots,” Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), pp.1322–1328, 1999.
[5] S. Thrun, W. Burgard and D. Fox: Probabilistic Robotics. The MIT Press, 2005.（邦訳）上田 隆一：確率ロボティクス．マイナビ，2007.
[6] D. Fox: “Adapting the Sample Size in Particle Filters Through KLD-Sampling,” Int. J. of Robotics Research, vol.22, no.12, pp.985–1003, 2003.
[7] D. Fox, W. Burgard, F. Dellaert and S. Thrun: “Monte Carlo Localization: Efficient Position Estimation for Mobile Robots,” Proc. of AAAI Conf. on Artificial Intelligence, pp.343–349, 1999.
[8] K. Murphy and S. Russell: Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. Arnaud Doucet, et al. Ed., Sequential Monte Carlo Methods in Practice. Springer, 2001.
[9] G. Grisetti, C. Stachniss and W. Burgard: “Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters,” IEEE Trans. on Robotics, vol.23, no.1, pp.34–46, 2007.
[10] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit: “FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges,” Proc. of Int. Joint Conf. on Artificial Intelligence (IJCAI), pp.1151–1156. 2003.
[11] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit: “FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem,” Proc. of AAAI Conf. on Artificial Intelligence, pp.593–598, 2002.
[12] D. Hahnel, W. Burgard, D. Fox and S. Thrun: “An Efficient FastSLAM Algorithm for Generating Maps of Large-Scale Cyclic Environments from Raw Laser Range Measurements,” Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp.206–211, 2003.
[13] S. Thrun, D. Fox and W. Burgard: “Monte Carlo Localization with Mixture Proposal Distribution,” Proc. of AAAI Conf. on Artificial Intelligence, pp.859–865, 2000.
[14] 原祥尭：“自己位置推定・地図生成（SLAM）の全体像”，連載：SLAM とは何か，日経 Robotics，2016 年 5 月号，pp.14–16, 2016.
[15] 原祥尭，坪内孝司，大島章：“確率的に蓄積したスキャン形状により過去を考慮した Rao-Blackwellized Particle Filter SLAM”，日本機械学会論文集，vol.82, no.834, pp.1–21 (DOI: 10.1299/transjsme.15-00421), 2016.
[16] O. Brock and O. Khatib: “High-Speed Navigation Using the Global Dynamic Window Approach,” Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), pp.341–346, 1999.
[17] E. Rimon and D.E. Koditschek: “Exact Robot Navigation Using Artificial Potential Functions,” IEEE Trans. on Robotics and Automation, vol.8, no.5, pp.501–518, 1992.
[18] D. Fox, W. Burgard and S. Thrun: “The Dynamic Window Approach to Collision Avoidance,” IEEE Robotics and Automation Mag., vol.4, no.1, pp.23–33, 1997.
[19] 友納正裕：“移動ロボットの環境認識—地図構築と自己位置推定”，システム制御情報学会誌，vol.60, no.12, pp.509–514, 2016.
[20] W. Hess, D. Kohler, H. Rapp and D. Andor: “Real-Time Loop Closure in 2D LIDAR SLAM,” Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), pp.1271–1278, 2016.
[21] 友納正裕：“2D レーザスキャナによる SLAM における地図の点群表現とループ閉じ込み”，第 22 回ロボティクスシンポジア予稿集，pp.42–47, 2017.
[22] つくばチャレンジ，http://www.tsukubachallenge.jp/
[23] 原祥尭：ROS の活用による屋外の歩行者空間に適応した自律移動ロボットの開発，第 94 回ロボット工学セミナー講演資料，2015. https://www.slideshare.net/hara-y/ros-slam-navigation-rsj-seminar
[24] 原祥尭：ROS を用いた自律移動ロボットのシステム構築，第 99 回ロボット工学セミナー講演資料，2016. https://www.slideshare.net/hara-y/ros-nav-rsj-seminar

原 祥尭（Yoshitaka Hara）

2007 年筑波大学大学院システム情報工学研究科博士前期課程修了．2007～2011 年日立製作所日立研究所（旧，機械研究所）にて，物流支援ロボットや搭乗型移動支援ロボットなどの研究開発に従事．2012～2015 年日本学術振興会特別研究員 DC1．2015 年筑波大学大学院システム情報工学研究科博士後期課程修了，博士（工学）．同年より千葉工業大学未来ロボット技術研究センター（fuRo）主任研究員．自律走行や物体認識に関する研究に従事．計測自動制御学会，IEEE の会員． （日本ロボット学会正会員）