fuRo 内製技術の統合による自律移動ロボットの開発と実証実験

○原 祥尭 [†1]，入江 清 [†1]，吉田 智章 [†1]，西村 健志 [†1]，大和 秀彰 [†1]，友納 正裕 [†1]

Development and Experiment of an Autonomous Mobile Robot by Integration of fuRo In-house Technology

*Yoshitaka HARA [‡1], Kiyoshi IRIE [‡1], Tomoaki YOSHIDA [‡1],
Takeshi NISHIMURA [‡1], Hideaki YAMATO [‡1], Masahiro TOMONO [‡1]

千葉工業大学 fuRo アウトドア部 II

1. 緒言

これまで fuRo では、多様な移動ロボットを開発すると共に、内製による基盤技術を蓄積してきた。ここ数年はつくばチャレンジに開発リソースを確保できなかったが、本年は実証実験の場として、fuRo のハードウェアおよびソフトウェアの内製技術を統合し、自律移動システムの開発を行った。

7 月下旬に全体方針を決定し、そこから 3 ヶ月程度でハードとソフトを組み上げた。時間不足のため課題は残るが、システム全体を実装でき、2 km の完走を達成した。本稿では、開発した自律移動ロボットと走行実験の結果について報告する。

2. ハードウェア

2.1 車体と搭載デバイス

Fig. 1 に、開発した自律移動ロボット Puffin の外観を示す。搭載しているセンサは、3D-LIDAR（Velodyne VLP-16）、IMU（Xsens MTi-3）、モータエンコーダ（後述のロータ角センサ）というシンプルな構成である。

車体は、アルミフレームによる拡張が容易な構造としつつ、全体を樹脂カバーで覆い防水性能を確保した。電源には、リチウムイオンバッテリ（IDX E-HL9S、14.4 V、87 Wh）を 3 直列 2 並列で搭載している。将来的に組み込み PC の電源供給も想定した大容量になっており、駆動系とセンサ系だけならばバッテリ交換なしで一日中走行できた。

コンピュータは、実験の利便性からノート PC（Lenovo ThinkPad P51）を使用した。バッテリ駆動時間は、処理負荷や画面輝度によるが 3 時間程度であり、一日あたり 1〜2 回のバッテリ交換を行った。

なお車体上部には、視認性を高めるためにパトランプを設置した。小型のロボットは生垣などの影に隠れやすく、歩行者や自転車から見て気付かれない場合もある。一般の通行人がいる環境での実験において、安全を確保するためには視認性も重要だと考えている。

2.2 駆動ユニット

車輪の駆動には、fuRo にて開発を進めているインホイール型ユニットを使用した。駆動ユニットの外観を **Fig. 2** に、仕様を **Table 1** に示す。

駆動ユニットは、モータ（ステータ・ロータ）、ロータ角センサ、減速機から構成される。いずれの構成要素においても専用設計を行っており、既製品の組み合わせでは難しい小型・軽量化と機能最適化を追求している。

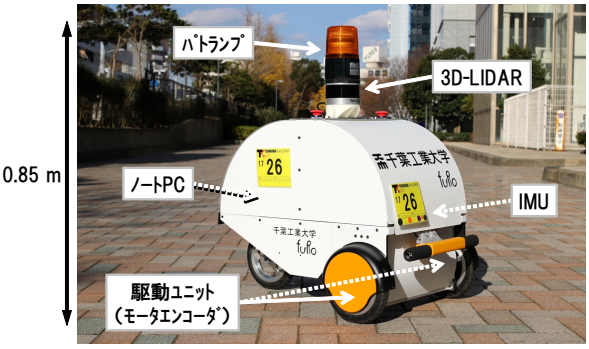

Fig. 1 自律移動ロボット Puffin

Fig. 2 駆動ユニット（モータ、減速機）

Table 1 駆動ユニットの仕様

全体サイズ	ϕ 70 mm、D 60 mm、940 g
モータ	3 相ブラシレス（5 極対）方式
ステータ・ロータ	ϕ 60 mm、積厚 20 mm
熱定格、トルク	400 W、0.93 Nm/10Arms
ロータ角センサ	磁気ディスク & ホール IC 方式
減速機	サイクロイド方式
速比	1/15（効率 80% 以上）
出力	フランジ式クロスローラ軸受

またモータコントローラ（ドライバ）は、本実装では連続電流 15 Arms、最大電流 30 Arms である。ノート PC が車体速度を統括し、CANopen の通信により左右輪の各々で速度制御する。

3. ソフトウェア

実装には、C++ と Python を用いている。各デバイス通信・制御プロセスや認識プロセス、要素機能ライブラリは、C++ でコーディングした。上位の行動決定プロセスは Python で実装し、SWIG 経由で C++ ライブラリを呼び出す構成とした。

プロセス間通信（IPC）には、独自開発の fuRom [1] を使用する。ROS [2] と類似した API で、共有メモリによる低遅延

[†1] 千葉工業大学 未来ロボット技術研究センター（fuRo）
[‡1] Future Robotics Technology Center, Chiba Institute of Technology

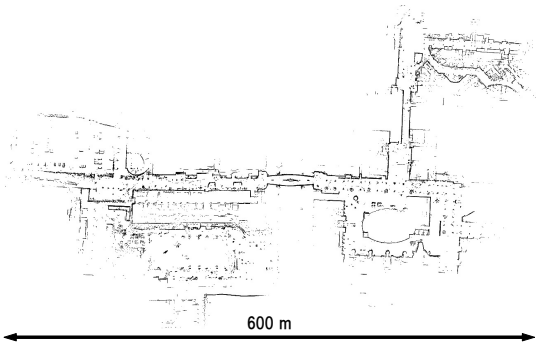

Fig. 3　SLAM で生成した 2 次元点群地図

な通信が可能なミドルウェアである。また、通信データの可視化（ビューア）や通信データの記録と再生（レコーダ・プレーヤ）などのツール群により、効率の良い開発ができる。

基礎となる走行制御は、差動駆動のキネマティクス（FK、IK）と直線経路追従を実装した。またオドメトリとして、車体静止時の IMU 自動キャリブレーション機能を持つジャイロ併用車輪オドメトリを実装した。

本システムでは、事前に手動走行によってセンサデータを記録し、オフラインで地図生成（SLAM）と経路設定を行う。現状は、SLAM の結果をオペレータが確認して成否を判断し、自律走行に使用する地図を決定している。人手による地図の修正は行っていない。また自律走行の目標経路は、手動走行時の経路を基本としつつ、人手で微修正して設定している。この作業のため、GUI の経路エディタを作成した。

SLAM には、スキャンマッチング（フロントエンド）と Graph-Based SLAM（バックエンド）による手法 [3] を用いた。今回は、水平な路面を仮定して 2 次元平面で行う。3D-LIDAR（Velodyne VLP-16）の 1 レイヤのスキャン点群を投影し、2 次元スキャンとして扱っている。ポーズグラフの最適化には、線形代数ライブラリ Eigen のみに依存した実装 [4] を使用した。**Fig. 3** に、生成した 2 次元点群地図を示す。

自律走行時の自己位置推定は、Particle Filter による Monte Carlo Localization で行う。ただし本実装の特徴として、占有格子地図ではなく、点群地図を用いている。すなわち、SLAM で生成した 2 次元点群地図と現在のスキャン点群を直接照合し、各パーティクルの重みを計算している。

障害物検出は、3D-LIDAR の 3 次元スキャンから衝突する高さの点群を抽出している。また最短測定距離（0.5 m 程度）未満にある障害物を検出するため、簡易的な路面検出を行い、路面がない場合も障害物として扱う。ロボットに人などが接近した際の安全確保には、この機能が重要であった。

経路計画は、基本的には事前に設定した経路（折れ線）に追従して走行し、目標経路上に障害物がある場合には減速・停止する。所定の時間が経過しても経路が塞がれている場合は、局所的な占有格子地図を生成して回避行動を行う。占有格子地図において障害物からの距離をコストとしてグラフを作成し、ダイクストラ法で経路計画する。

Fig. 4 に、自律走行時のビューア画面を示す。地図にレーザスキャンが重なっており、ロボット位置のパーティクルは正確な推定ができている。一方でオドメトリの位置は、若干ずれていることが分かる。ロボットは、経路に沿って自律走行

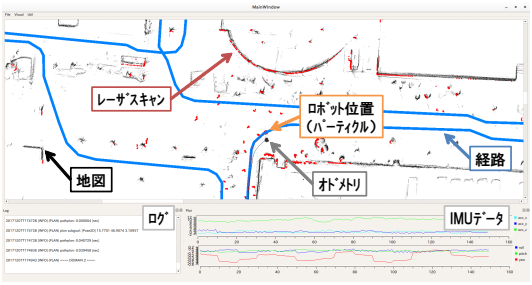

Fig. 4　自律走行時の可視化

Fig. 5　11/3(金) 自律走行時の人混みの様子

している。ビューア下部には、ログのメッセージや IMU データのグラフを表示している。

4.　実験記録

実験には、10/14(土)、11/3(金)、11/4(土)、11/5(日) の 4 回参加した。10/14(土) は、手動走行によるデータ取得のみを行い、11/3(金) 以降に自律走行を行った。

11/3(金) は、地図生成のために手動走行を 2 回行い、その後は 10/14(土) の地図で自律走行をテストした。落ち葉や木の実の上を走行しないように経路を修正し、大清水公園の確認走行を達成できた。全区間の自律走行も 2 回試行したが、センター広場でイベントによる人混みが発生しており、またこの時点では障害物回避を統合していなかったため、通過できなかった。**Fig. 5** に、人混みの様子を示す。経路が完全に塞がれ、ロボットが通る隙間もないような状況だった。

11/4(土) は、前日 11/3(金) の地図で自律走行をテストした。障害物回避に不具合があったが、回避が発生しなければ全区間を走破できた。5 回の試行で、2 回は障害物を回避後に目標経路に復帰できなかったが、3 回は一応完走できた。ただし、折り返し地点付近のコースを誤ってショートカットして設定していたことが発覚し、目標経路の再設定を行った。

11/5(日) は、本走行前の調整走行で、初めて障害物回避ありで正しいコースの 2 km を完走した。ただし、順番待ちを避けてスタート地点の先から走行開始した。本走行では、2 番目の出走でスタート付近に人が多かったが、完走を達成できた。

5.　結言

本稿では、fuRo の内製技術を統合して開発した自律移動ロボットと、つくばチャレンジでの走行実験の結果について述べた。今後は、ハードとソフトの両面で、改良を継続する。

参考文献

[1] 入江 清: “ROS との相互運用性に配慮した共有メモリによる低遅延プロセス間通信フレームワーク”, 日本ロボット学会学術講演会予稿集, 2017.
[2] ROS. `http://www.ros.org/`
[3] 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, 2017.
[4] Kiyoshi Irie and Masahiro Tomono: “A Compact and Portable Implementation of Graph-Based SLAM,” *Proc. of JSME Robotics and Mechatronics Conf. (ROBOMECH)*, 2017.