ROSで動作するロボットをクラウドで管理し任意のノードを実行可能なWebシステムの開発と応用

○末永良太 (明治大学) 森岡一幸 (明治大学)

Development and application of a web system that manages ROS-operated robots in the cloud and runs arbitrary nodes

○ Ryota SUENAGA(Meiji University), and Kazuyuki MORIOKA(Meiji University)

Abstract : This paper presents a web-based robot management system named Rowma(Robot Web Manager) which monitors and controls ROS based robots near the user via a web browser. This system consists of 3 components: a ROS package, a WebSocket server for robots management, and an npm package used for a web client. The ROS package collects ROS information, accesses to the server, and executes launch files selected from users. In this paper, details of the proposed system are provided and an application using the proposed system is introduced as an example.

1. 緒言

掃除ロボットのような一般に広く普及していて認知度も高いロボットだけではなく、ロボット技術が活用されたシステムはこれから身近に増えてくるだろう。掃除ロボットのように単機能で特定の作業を任せるものだけでなく、複数の作業を行える汎用的なものの必要性も増えてくることが予想される。さらには、スマートフォンやスマートスピーカーのように、新たなアプリをインストールして実行したりすることも考えられる。この実現のためにはロボット開発者がプログラムを実行させるのではなくユーザーがPCやスマートフォンから状況に応じてロボットのROSシステムを操作できるような環境が必要になる。この環境のためのインターフェイスもまた重要である。現在でもroslibjsなどを使用することによってブラウザを介してROSのトピックやサービスの取得や送信が可能であるが、事前に決められた特定のロボットにて起動中の特定のノードに対して作用させることしかできない。つまり既存のものでは様々なアプリ（ROSのノード）を持つロボットにて、タスクに応じて任意のノードを起動してロボットに新たな機能をもたせたりすることは難しい。更に、付近に存在する様々なロボットの中から特定の1台を指定して、そのロボットが実行可能な処理を選んだ上で実行させる必要がある。これらの処理を全てブラウザから操作できるようにするシステムがあればROSベースのシステムを一般ユーザーが活用するためのロボットアプリケーションを容易に構築できるようになると考える。

そこでこの論文では、ROSベースのロボットの情報を管理し、ユーザーがWebブラウザにて付近に存在するロボットを検索して、選択したロボットに対してroslaunchを実行することが出来るシステムであるROS Web Manager(Rowma)について述べる。本論文では位置情報と共にロボットの接続を管理するサーバーソフトウェアと位置情報を含めてサーバーへ接続を行う機能を持つROSノード、そしてWebブラウザからサーバーへ接続するためのライブラリを開発した。

本論文の第2章ではシステム全体の構成、第3章では実際に開発したシステムに関する説明、第4章では開発したシステムの応用に関してを述べ、最後に第5章でこれらの内容をまとめる。

2. ロボット管理システム

ROS Web Manager(Rowma)とは、ROSベースのソフトウェア構成を持つロボットをクラウドで一元管理し、ユーザーがWebブラウザから付近に存在する特定のロボットに対してrosrunやroslaunchなどROSに関係する任意の処理を行わせることや自動で最新版のソースコードを適用することなどを可能にするシステムである。Fig. 1はこのシステムの概要である。ユーザーはWebブラウザを通じて付近のロボットの確認と操作対象ロボットの選択、操作対象ロボットに対するroslaunchコマンドの実行が可能である。

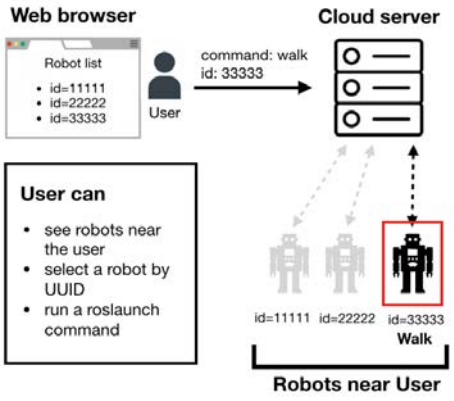

Fig. 1: Overview of Rowma

SY0010/19/0000 - 0920

本システムを使用することにより、インターネット環境さえあればどこからでも対象ロボットの ROS システムに存在する任意の ROS パッケージのノードを実行することができる。また、今後の実装により実行だけではなくパッケージの新規追加や修正などの更新作業をブラウザ上で行えるようにすることも可能である。

ロボットをインターネット経由で遠隔地から操作する研究は、古くは 1990 年代から行われている [1]。近年ではブラウザでのロボット開発の事例として AWS RoboMaker[2] や、RT コンポーネントを用いたシステム構築と実行をブラウザ上で行うことができる RT システムエディタ on the Web[3] なども存在する。

特に、AWS RoboMaker はブラウザ上で動く ROS ソフトウェア提供しており、Greengrass というミドルウェアを適切に設定して使うことでブラウザからのロボット操作を実現している。これにより、ブラウザ上で ROS の各サービスを使用してロボットアプリケーションを作成し、ロボットに Greengrass を組み込み AWS 側で設定を行うことでアプリケーション作成から実機のアップデートまでブラウザのみで行うことが出来る。

本論文で開発するシステムは既存の ROS システムに今回開発したパッケージを追加するだけで良く、ロボットに対する操作に関しても npm パッケージとして API を提供しているので、汎用性が高く既存の ROS システムへ容易に組み込むことができるという利点がある。

3. 開発したシステム

3.1 システム概要

本システムはロボットの接続管理用 WebSocket サーバーと WebSocket クライアントとなる管理用 ROS パッケージ、Web ブラウザでロボットに対する操作を可能にするための管理用 npm パッケージの 3 つで構成されている。以下の Fig. 2 はシステムの概要図である。まず最初に ROS システムにインストールされた管理用 ROS パッケージ中のスクリプトを実行すると UUID を生成して位置情報と共に管理用 WebSocket サーバーへ接続を行う。管理用 npm パッケージが組み込まれた Web ページを Web ブラウザで表示し、付近のロボットの接続情報を取得して、該当ロボットの UUID からロボットを選択し実行したいコマンドを指定して管理用 WebSocket サーバーを介してロボットにそのコマンドを実行させる。

3.2 管理用 ROS パッケージ

管理用 ROS パッケージは、起動時に生成した UUID と位置情報、roslaunch で実行可能な launch ファイル一覧と

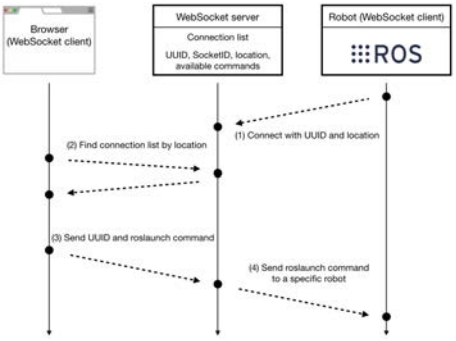

Fig. 2: Overview of the system

いった、ロボットに関する情報を WebSocket サーバー接続時に送信して、その後は通信を待ち受ける WebSocket クライアントを含むものである。この管理用 ROS パッケージ内のスクリプトを起動後、WebSocket サーバーからの通信を待ち続けて、サーバーから roslaunch 実行の通信を受け取った時にそれを実行する。なお、WebSocket クライアントは接続時に固有の SocketID を発行し、WebSocket サーバーはこの SocketID を指定して通信することで任意の端末と通信することが出来る。また、実行可能な launch ファイルの一覧は管理用 ROS パッケージ内のスクリプトにて `find` と `grep` をシェルコマンドとして実行することにより取得する。

3.3 WebSocket サーバー

WebSocket サーバーは socket.io という JavaScript のライブラリを使用して AWS EC2 上で構築され、管理用 ROS パッケージからの接続情報をプログラム中に保持するものである。この WebSocket サーバーは位置情報を入力として接続情報を保持した配列からその位置情報を持つロボット一覧を返す API と、UUID と実行コマンドを入力として該当 UUID のロボットに対して指定されたコマンドを送信する機能を提供している。

3.4 管理用 npm パッケージ

管理用 npm パッケージとは Node.js で使用可能なライブラリなどのことで、Web ページの構築にあたって容易に使用することが出来る。本システムの管理用 npm パッケージはブラウザから WebSocket サーバーと通信するためのライブラリで、位置情報をパラメーターとしてサーバーに付近の接続リストを要求するリクエストを送信する関数と WebSocket サーバーに接続して UUID と実行コマンドをパラメーターとした WebSocket 通信を送信する関数から構成される。

3.5 処理の流れ

開発したシステムには大きく分けて 3 つの処理が存在する。1 つ目は管理用 ROS パッケージと WebSocket を接続する処理、2 つ目は Web ブラウザで位置情報から付近のロボット一覧を取得する処理、そして 3 つ目は Web ブラウザからロボットに対して roslaunch を実行する処理である。この節では図を用いてそれぞれの処理について説明する。

Fig. 3 に管理用 ROS パッケージが WebSocket サーバーに自身の情報を送信して接続する流れを示す。まず管理用 ROS パッケージは UUID を生成し、その後 freegeoip.app を使用して IP アドレスから現在の緯度経度を求めこれを Geohash という文字列 (後述) に変換する。そしてその端末が実行可能な roslaunch コマンドを取得した後、WebSocket サーバーに接続しこれらのデータを送信する。

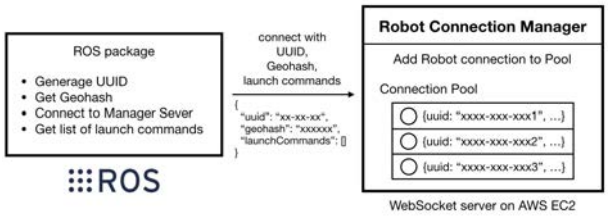

Fig. 3: Overview of the server connection

Fig. 4 に管理用 npm パッケージが WebSocket サーバーから付近のロボットの一覧を取得する流れを示す。管理用 npm パッケージも管理用 ROS パッケージと同様に freegeoip.app を利用してユーザーの現在地の緯度経度を求めてこれを Geohash に変換する。そしてこの Geohash を用いて WebSocket サーバーに接続し、同じ Geohash を持つロボット一覧を取得する。

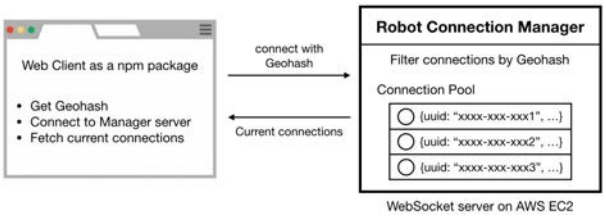

Fig. 4: Overview of the robots list

Fig. 5 にブラウザでロボットと実行コマンドを指定してロボットにそのコマンドを実行させる流れを示す。まず取得したロボット一覧からコマンドを実行したいロボットの UUID と実際に実行するコマンドを選択して管理用 npm パッケージを組み込んだ Web ページから WebSocket 通信を実行する。これを受け取った WebSocket サーバーは保存してある接続情報から該当 UUID を持つロボットを検索し、このロボットに対してコマンドを実行するための WebSocket 通信を実行する。

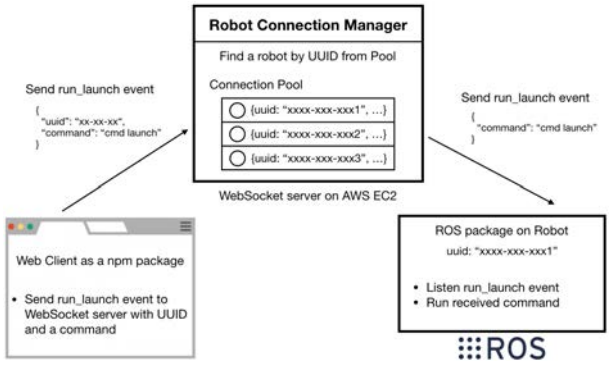

Fig. 5: Overview of the command execution

Fig. 6 は管理用 npm パッケージを使用したサンプルアプリケーションである。画面上の一番左のボタンを押すと WebSocket サーバーへ接続するために現在地の Geohash を取得し、隣のボタンを押すと WebSocket サーバーから接続中のロボット一覧を取得してドロップダウン内に選択可能な値が入る。そして実行対象のロボットと実行コマンドを選択して一番右のボタンを押すと実際にロボット上で指定したコマンドが実行されるものである。

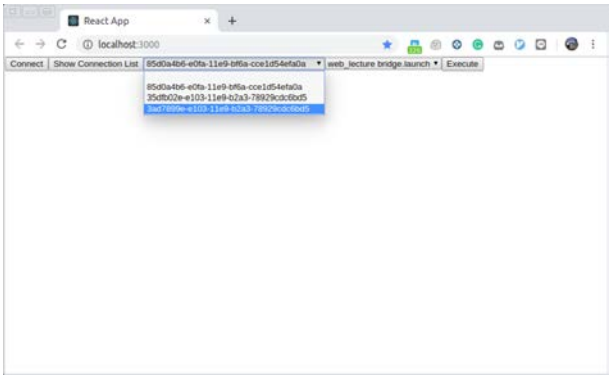

Fig. 6: Sample application

3.6 サーバーに保持するデータの構造

WebSocket サーバーには以下の Table. 1 で示す構造のデータが格納されている。UUID は管理用 ROS パッケージ中で Python の uuid ライブラリから生成された、システム全体で使用するロボットの識別子であり、Socket ID は WebSocket の接続毎に自動で生成される接続固有の識別子である。また、Geohash はロボットの現在地であり、Commands はそのロボットが実行可能な roslaunch コマンドの配列である。

3.7 位置情報の取得

本システムでは付近のロボットの位置を取得するために Geohash を用いる。Geohash とは位置の管理に地球上の座

Table 1: Data structure

Column	Type
UUID	String
Socket ID	String
Geohash	String
Commands	Array(String)

標を緯度経度に基づいて分割して緯度経度の数値を 1 つの文字列として表したもので、例えば本システムの動作確認を行った明治大学中野キャンパスの Geohash は `xn774h0` となる。

4. ロボット管理システムの応用

本章では今回開発したユーザーの近くに存在する ROS ベースのロボットシステムにおいて launch ファイルを Web ブラウザから実行できるシステムを応用した具体的なシステムとして、深層強化学習に基づく車輪型移動ロボットの自律走行システムの学習と開発を支援する Web システムについて述べる。

4.1 概要

まずは、現在開発中の Web ベースの深層強化学習に基づく移動ロボットの自律走行システムの開発と仕組みの学習を支援する教育システムの rourse の概要について説明する。

近年、深層強化学習によりゲームなどのタスクにおいて人間を超えるスキルを獲得する例が多数報告されている。移動ロボットの自律走行システムへの応用も盛んに取り組まれており、我々も DDQN に基づいた行動学習を移動ロボットの自律走行システムに適用して、リアルワールドでの走行システムを開発している [4]。rourse はこのような深層強化学習に基づいた行動学習と車輪型移動ロボットの自律走行システムの開発を通じて、移動ロボットのシステムに関してユーザーが学ぶことのできる Web システムである。ユーザーは Web ブラウザ上で走行させたい環境の二次元地図を与え、強化学習やロボットのパラメータを調整し、クラウド上の学習プログラムを動作させて、ロボットの行動モデルを獲得することが可能になっている。得られた行動モデルを用いて実環境での自律走行システムとして動作させるには、実際の移動ロボットに行動モデルを適用して、走行プログラムを起動する必要がある。ロボットにログインして ROS のノードや launch ファイルを実行すればそれは可能になるが、前述した行動学習と同様に、ロボットの走行に関しても Web ブラウザからの操作で完結させることで、ロボットシステム開発に取り組んだ経験が少ないユーザーでも、深層強化学習によるロボット走行システムの開発の一連の流れを体験できるようになる。この Web ブラウザから実際の移動ロボットでの走行のための launch ファイルの実行において、本稿で提案したロボット管理システムを適用する。Fig. 7 は rourse のシステム構成図である。

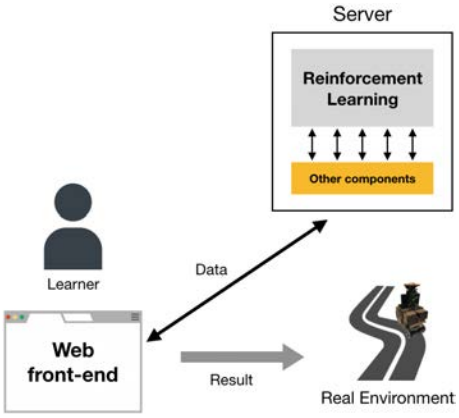

Fig. 7: The overview of the system

ユーザーは Fig. 8 に示すような Web ページ上で深層強化学習やロボット制御のための各種パラメーターについて、アニメーションによりその効果を確認しながらパラメーターを設定する。設定したパラメーターにより、クラウド上のサーバーにて行動モデルの学習を実行する。学習の経過は Fig. 9 に示すように、Web ブラウザ上に表示されるシミュレーター上で確認することができ、学習終了時には学習済みモデルをダウンロードすることが出来る。そしてダウンロードしたモデルを使って実際にロボットを自律走行させ、学習や制御のパラメーターの意味や効果を学び、強化学習やロボット工学に関する知識を得ることがこのシステムの目的である。

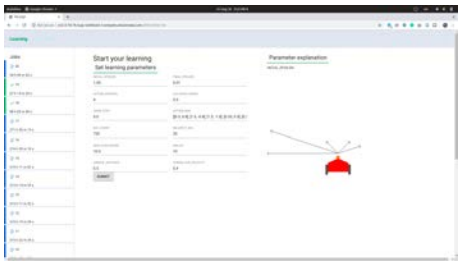

Fig. 8: Robot parameter page of Rourse

このように、このシステムはインターネット環境さえあれば強化学習によって生成した行動決定モデルを使って誰でも容易にロボットを実環境で自律走行させることができ、強化学習やロボット工学に関する理解も深めやすくなることが期待できる。

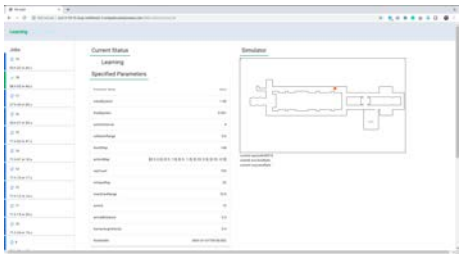

Fig. 9: Learning status page of Rourse

Table 2: Configurable parameters for the proposed learning system.

TRAIN	学習するかどうかのフラグ
INITIAL_EPSILON	ランダム行動の初期確率
FINAL_EPSILON	ランダム行動確率が減ったあとの最終的な確率
ACTION_INTERVAL	行動選択する時間間隔
ACTION_MAP	選択する行動
NN_INPUT_RAY	学習に入力するためのレーザー測定点数
MAX_SCAN_RANGE	レーザーセンサで見る最大距離
ARRIVAL_DISTANCE	目的地に到着したかどうかの判断をする距離
REWARD	報酬
STATE	状態

4.2 深層強化学習を用いた自律移動ロボットの自律走行

深層強化学習による自律移動ロボットは、一般的に走行時のセンサ情報などを状態として、学習済みの行動モデルから状態に対応する走行制御命令を行動として出力し、それを繰り返すことで走行する。例えば、我々の過去の研究 [4] では、2D-LiDAR によるロボット周囲の障害物までの距離情報とウェイポイントまでの相対位置姿勢を状態とし、直進や左折、右折に対応する並進速度と回転速度の速度指令値のセットを出力としている。行動モデルの学習時の報酬として、ロボット周囲の障害物の有無、ゴールへの到達や壁との衝突、ゴールへの接近状態に対して、適切な正負の値を設定することで、行動モデルの学習が可能であることも示されている。rourse ではこのシステムに基づいて、学習やロボットの行動のパラメーターを Web ブラウザ上で設定可能としている。Table. 2 に学習で設定可能なパラメーターの例を示す。

自律移動ロボットの行動モデルの学習は、独自に開発した Fabot2D というシミュレーターを用いた深層強化学習で行う。このシミュレーターは任意の環境の地図情報を読み込み、任意の人数の歩行者を配置することが可能である。このシミュレーターをクラウド上で実行して前述した状態と行動、報酬に基づいた学習を行い、シミュレーター環境内で壁と歩行者に衝突せずに目的地まで走行できるようにする。学習中のロボットの状態を確認するために 9 の右側にあるような、クラウドで学習中の Fabot2D の地図やロボット位置と同じものを Web ブラウザ上で表示する Web シミュレーターを作成した。

4.3 ロボット管理システムを適用した実際の移動ロボットの走行

rourse において、Web ブラウザ上から実ロボットへの学習済み行動モデルの適用と自律走行の launch ファイルの実行までをシームレスに行うためにロボット管理ツールを組み込む。今回はレーザーセンサ用ノード、移動ロボットの制御ノード学習済みモデルを読み込み速度指令値を出力するノードなど、自律走行に必要なノードをまとめて実行するための launch ファイルを、Web ブラウザから起動することで自律走行システムが実現可能である。実際に北陽電機の 2D-LiDAR を搭載した i-cart mini ベースの移動ロボットを用いて、提案システムを使った自律走行を実現している。

この管理システムを組み込む事でユーザーは Web ブラウザとロボット用 PC とを行き来して操作することなくモデルの学習からロボットへのモデルの適用と自律走行の実行までを Web ブラウザ上で完結することができる。

5. 結言

本論文では、ROS ベースのロボットシステムを管理し、Web ブラウザからユーザーの付近に存在するロボットを選択して launch ファイルを実行することができるロボット管理システムについての開発と、現在開発されている移動ロボットの自律走行を学習するシステムへの実際の応用に関して述べた。

現在は UUID によりロボットを選択し、ROS システムに保存されている launch ファイル名をそのまま表示して実行しているが、ユーザーの使い勝手を向上させるためには、対応するラベルを付与してそれをブラウザにて表示できるようにするなどの改善が必要になる。なお、今回開発したシステムのソースコードは GitHub にアップロードされており、誰でも閲覧可能である。

参考文献

[1] Goldberg, Ken, et al. ”Desktop teleoperation via the world wide web.” Proceedings of 1995 IEEE International Conference on Robotics and Automation. Vol. 1. IEEE, 1995.

[2] ”AWS RoboMaker”, `https://aws.amazon.com/robomaker/`.

[3] ”RT システムエディタ on the Web”, `http://hara.jpn.com/_default/ja/Software/RTSEoW.html`.

[4] Yuki Kato and Kazuyuki Morioka, ”Autonomous Robot Navigation System Without Grid Maps Based on Double Deep Q-Network and RTK-GNSS Localization in Outdoor Environments”, Proceedings of IEEE/SICE International Symposium on System Integration (SII 2019), pp.346–351, 2019.