解説

ROS (ロボット・オペレーティング・システム)

ROS (Robot Operating System)

岡田 慧* *東京大学大学院情報理工学系研究科

Kei Okada* *The University of Tokyo, Graduate School of Information Science and Technology

1. はじめに

ROS (Robot Operating System) と呼ばれるロボットソフトウェアのプロジェクトが欧米を中心に急速にユーザーを惹き付け注目を集めている [1]. 図 1 に ROS を利用している 150 強の研究機関や個人の分布 [2] を示すが、米国だけでなく欧州でも盛んに利用されており、アジアやオーストラリアでも広がっている。また、ロボット関連の国際学会においても ROS に関連するワークショップや付随会議が開催され多くの参加者を集めており関心の高さを伺わせる。

筆者の研究室でも ROS を使ったオープンソースプロジェクトに参画しており、図 2 に示すように他の研究機関で開発されたソースコードを利用しつつ、我々も開発したソースコードをオープンにしながら、海外の研究機関らと共同で研究を進めてきている [3].

正直な感想としては、これほど有用なソフトウェアは存在せず、ROS を使う前と使う後では世界が一変したと感じている。例えば、図 2 の実験では 7F の研究室から 2F のサンドイッチ売り場までロボットの自律移動を行っているが、オープンになっているプログラムをダウンロードして起動すればすぐに自律移動できる状態にあり、我々はその上でエレベータの認識や昇降、店員とのインタラクションの研究に注力すればよい。さらに、この研究ではドイツのミュンヘン工科大学の Lars Kunze 氏との共同研究であり、彼らのシステムは Prolog を使い、我々のプログラムは C++ と Lisp を用いたものであったが、ROS という共通言語・ツールを活用することでスムーズな連携が実現できた。

ロボットの研究用のソフトウェアプロジェクトは他にも存在するが、ROS は突出して大量・高品質の再利用可能な知能ロボットモジュール群と、これらをスムーズに連携可能なツール群を提供しているように見える。本稿ではその背景を探り、さらにこのようなプロジェクトに対してどのような対応が考えられるのか、私見を述べてみたい。

図 1 ROS を活用している研究機関の分布

図 2 東大 JSK での ROS を使った知能ロボット研究の例

2. ROS の特徴

ROS はロボット・オペレーティング・システムの略であるが、この言葉から ROS の実態について正確に把握するのは難しい。実際に ROS ユーザーの中でも混乱が生じていることが見られ、後述する ROS 質問サイトでも「ROS とは何か？ ミドルウェア？ フレームワーク？ オペレーティングシステム？ (What is ROS exactly? Middleware, Framework, Operating System?)」という質問が投げかけられている。これに対し ROS プロジェクトの中心人物 Brian Garkey 博士は、「ROS = 通信ライブラリ + ツール + 基盤 + エコシステム」と答え、それぞれ以下の様に説明している [4].

通信ライブラリ：ROS は分散型計算システムの迅速、簡易な構築のために設計された出版・購読型のメッセー

原稿受付 2012 年 9 月 25 日

キーワード：Open Source Robotics, Software Platform, Open Community

*〒 113-8656 文京区本郷 7-3-1

*Bunkyo-ku, Tokyo

ジ通信基盤を提供するものである。

ツール：ROS は分散型計算システムの設定、起動、監視、デバッグ、視覚化、ログ取り、停止を行う広範囲なツールを提供するものである。

基盤：ROS は移動、操作、認識を中心に大量の有用なロボットライブラリ群を提供するものである。

エコシステム (コミュニティ)：ROS はインテグレーションとドキュメンテーションを中心として大規模なコミュニティによって支えられ、発展している。ros.org は世界中の開発者から提供された大量の ROS パッケージを見つけ、習得するためのワンストップサービスである通信ライブラリはいわゆるミドルウェアのレイヤーであり、また基盤とされているものは一般に知能ロボット用ライブラリ・モジュールと呼ばれるものであろう。これらは他のロボットのソフトウェアプロジェクトでもカバーされてるように見える。したがって他のプロジェクトと ROS の違いを探るとすると、残りの二つの項目であるツールとエコシステム (コミュニティ) に注目することになる。実際、これが ROS が広くユーザを集めている理由になっていると考えている。

2.1 パッケージマネージメントツール群

ROS は 2007 年の Stanford の AI 研の学生であった Morgan Quigley 氏が始めた並列分散型のロボットソフトウェアフレームワークである Switchyard プロジェクト [5] に起源があるが、ROS が他のロボットのソフトウェアプロジェクトに比べて多くのユーザを集めるようになる転換点は 2008 年にパーソナルロボット産業の育成を目的として創立された WillowGarage 社のサポートが加わり、Player/Stage プロジェクトの Brian Garkey 博士らも参加し、Switchyard プロジェクトが ROS [6] として組織化された時点であろう。

図 3 に WillowGarage 社の幹部が頻繁に利用するスライド [7] を紹介する。濃い部分は研究に必要な雑作業で薄い部分が研究そのものを示す。現状は上の状態で研究時間の多くはセンサやロボットから情報を取得するドライバを作成したり、結果を視覚化するツール開発等の雑作業に費やしており、ROS はこの状況を変えるために研究サポートを行うツールを提供することで、研究その物にかかる時間を増やし技術の発展を支えることを目的としている。

具体的なツールの例として、非階層型のファイルシステムを提供する package システムを中核に、パッケージが依存するすべてのパッケージを正しい順番で、かつ、必要なインクルードパスやライブラリのリンク情報を自動的に抽出しながらコンパイルするパッケージ依存関係自動解決コンパイルツール (rosmake)、複数のプログラムノードの実行、パラメータ設定、通信接続を行うプログラムノード実行管理ツール (roslaunch) を紹介する。

図 3 ROS のねらい:WillowGarage 社のスライドより [7]

2.1.1 パッケージとマニフェスト

ここでは自律地図作成や動作計画等のまとまった機能を提供するソフトウェア群が含まれるフォルダをパッケージ (package) と呼ぶ。ROS ツール内でのディレクトリ構造は指定されたパッケージが指し示すフォルダに、ディレクトリ階層を辿ることなく、直接移動できる非階層的なディレクトリ構造が導入されている。

また、各パッケージには manifest.xml と呼ばれるファイルを置き、そのパッケージが依存する他のパッケージや、そのパッケージをコンパイル・実行するために必要なライブラリを記述する。

2.1.2 パッケージ依存関係自動解決コンパイルツール

各パッケージには manifest.xml と呼ばれるファイルを置き、そのパッケージが依存する他のパッケージや、そのパッケージをコンパイル・実行するために必要なライブラリを記述することは前述した。これを使うことで、あるパッケージ A が依存するパッケージ B が存在したとき ROS システムはその依存関係を知ることができ、rosmake と呼ばれるコンパイルツールを用いてパッケージ A のコンパイルを指定すると、自動的にそのパッケージが依存するパッケージ B のコンパイルを行い、終了後にパッケージ A のコンパイルを行うような、パッケージの依存関係の解析に基づいたビルドが実現できる。

2.1.3 プログラムノード実行管理ツール

roslaunch と呼ばれる分散プロセスの起動をデータ通信の接続を行うツールは、パッケージ A の代表的な機能を実行する設定ファイルを a.launch に記述しパッケージ A のフォルダ内に置いておくとすると、roslaunch A a.launch とすることで、物理的なパッケージ A の場所にかかわらず ROS システムがパッケージ A の場所を見つけて a.launch の内容から必要なプログラムの起動と通信接続を行うようになっている。

2.1.4 ROS ツールの効用

これらのツールは地味だが大規模な知能ロボットソフトウェアを構築しようとした場合の効果は絶大である。従来

誰も編集可能な wiki
<http://www.ros.org/wiki/>

バグ報告、機能提案
<https://code.ros.org/trac>

質問共有サイト
<http://answers.ros.org/>

図 4 ROS におけるコミュニティサイトの例

はあるソフトウェアを利用しようとしたら、それが依存するライブラリやソフトウェアを逐一、各自の状況に応じてファイルや設定ファイルのパス等を変更しつつインストールし、さらにプログラムの実行時には複数のプログラムを一つずつ手順を踏んで実行する必要があった。ROS を使うことでこれが `rosmake`, `roslaunch` の 2 回のコマンド実行で済むようになる。実際に、1 章に示したドイツの学生との連携も、複雑なプログラムのコンパイル、起動手順を伝え合う必要はなく ROS を共通言語としてスムーズな連携が可能であったという背景がある。

またこれは別のシステム・環境で再現することも容易であることを意味する。従来のロボティクス研究では、各研究者が異なるシステム・環境を用いており実験結果の再現性や追試による評価など科学的なアプローチがなされてきていなかったが、ROS を活用することでこれを克服できる可能性がでてきている。誌面の都合上割愛したが、ROS では交換されるすべての情報を記録し再生する便利なツールも準備されており、これを活用することで、画像処理研究や SLAM 研究で見られるような、標準のデータセットを準備し世界中の研究者が再現・追試可能な状況で研究に取り組むことで急速に発展したのと同様の、科学的なアプローチによるロボティクス研究環境が構築されつつある。

実際にこの効果に気がついているプロジェクト推進側も出始めており、米国の最近の Challenge 系プロジェクトや、欧州 FP7 のが取り組んでいるロボットののための知識ネットワークを構築する RoboEarth [8] プロジェクトなどでは ROS の活用を義務付けている。ROS を利用することでパッケージ管理や、異なる研究機関でのコラボレーションが捗る点は前述したが、よい成果ができればそれはコミュニティに引き継がれ活用されているというメリットもあり、従来の様に大量の資金を投下して、論文は書くことができたが、ソフトウェア資産は引き継がれず、次のプロジェクトでは

再度同じようなソフトウェアを書くところから始める、という無駄はなく、効率的に成果を刈り取り、引き継げるようになっていくだろう。

2.2 オープンソースロボティクスと研究コミュニティ

ROS の特徴の一つにソースコードをすべて公開するオープンソースで開発を進めている点[†]があげられる[†]。しかしソースコードをオープンにするだけではロボットのソフトウェアプロジェクトは成功しない。文献 [9] における指摘を踏まえれば、オープンソースプロジェクトの要諦はソースコードをオープンにして誰でも入手できるようにすることで、世界中から共同開発者とアーリーアダプタを呼び込み研究開発コミュニティを構成し、そこでできるだけ多くの人の目でソースコードを確認してバグを出しつつ活発に議論をしながらスピード感をもってバージョンアップを繰り返し、短い期間で質の高いソフトウェアとさらなるユーザーの獲得を目指す戦略であり、ソースコードの公開は高品質のソフトウェアの開発とユーザー拡大のための手段でしかなく、ソフトウェアプロジェクトを支えるコミュニティが重要になってくる。

このコミュニティの大きさががソフトウェアの品質、すなわち知能ロボットのモジュールの頑健性ならびに再利用性といった品質を決定する点も注意したい。より多くの人が利用することで、開発者には思いもよらなかったバグや要件が見つかり、それをきちんと開発者にフィードバックすることで、次のバージョンでは修正され、よりよいソフトウェアになっていく。このサイクルを維持することなくでは、よいソフトウェアの実現はない。そして、安定した知能モジュールの存在は、その上でより高いレベルの研究を可能とし、ロボティクス研究の発展に大きく寄与する。図 2 の自律移動のプログラムは、この最も良い例であるが、世界中で活用されいくつものバグ報告や機能改善を経ること高い品質を実現し、これにより、我々は自律移動を前提としたより上位の研究に打ち込むことが可能となっている。また、我々が見つけたバグや機能提案は、半年ごとの ROS のシステムの更新でかなりの部分が取り込まれている。これにより、我々が開発した差分や変更点を自身で管理する

[†]ただし公開を義務付けているわけではない。非公開にしたければ選択が可能になっている。また公開されているほとんどのソフトは BSD ライセンスであるため、それを改変したものを公開する義務もない。

ことなく、開発者が管理してくれるため、自分たちは研究に没頭できる体制になっており、開発者へのフィードバックは大きなメリットがある。

また、ROS ではコミュニティを創出し維持するための仕組みとして図 4 に示すような利用者のためのソフトウェアパッケージに関する情報を共有する wiki ページ[†]や、開発者のためのソフトウェアパッケージの不具合報告や機能提案を行うためのバグトラッキングシステム^{††}、ROS やそれを使ったソフトウェアに関連する様々な質問と答えが集積される質問共有サイト^{†††}が存在する。

重要なのは、これらの wiki, バグサイト, 質問サイトは誰でもユーザ登録して編集できるようになっており、開発チームに籍があるかないか、あるいは関連するプロジェクトに参加しているか否かにかかわらず、誰でも同一の資格で開発者と利用者からなるコミュニティが形成され議論している点にある。また、ROS の技術的な方向性についても公開の会議やメーリングリスト[‡]で議論され規格を定めている。すなわち ROS は誰か個人や団体、企業の戦略の元に発展しているのではなく、利用者や開発者のコミュニティによってオープンに運営されており誰でも参加できるものになっている。

結果として、これらの仕組みは世界中で熱意をもって ROS を開発しより良くしていきたいという熱意のあるユーザがボランティア的に集結し活躍できる構造を提供しており、これにより ROS は急速にユーザを集め、かつソフトウェアを著しく進歩させることに成功していると言える。

ちなみに、最近のロボットの国際学会では若者が（時には年長者も）胸をはって ROS の T シャツを着て歩いている様子を見かけるが、彼らのほとんどは WillowGarage 社の社員ではなく ROS の関連会議などの出席し T シャツをもらった人たちであり、それがカッコがよいと信じて T シャツを着ている。このようなユーザの存在が ROS プロジェクトを大きくしてきており、彼らを見かける限り ROS プロジェクトは前に進んでいくと確信できる。

3. ROS のこれから

3.1 WillowGarage から OSRF へ

ROS は WillowGarage 社のサポートにより急拡大した

が、現在はその役割を終えつつあると判断し、新たに Brian Garkey 博士を代表とし、欧州やアジアの関係者を理事に迎えた Open Source Robotics Foundation (OSRF) と呼ばれるロボットのオープンソフトウェアの維持・管理する非営利財団を 2012 年 4 月に設立し、ここが ROS のサポートを引き継ぐこととなった。これにより、ROS は米国の 1 民間企業が主導しているプロジェクトから、名実ともに国際的な非営利組織が主導する、よりオープンでコミュニティの議論に基づいた運営がなされる体制となり、これまで以上に広い範囲で ROS を活用したプロジェクトが興ってくることが予想される。

3.2 オープンソースとロボット産業

ROS を活用したロボット産業も続々と生まれてきている。すでに WillowGarage 社からはテレプレゼンスロボットの販売を目指す Sutable Technologies、産業ロボットの視覚技術を提供する Industrial Perception、軽量安全マニピュレータを開発する RedWood Robotics、知能ロボットソフトのコンサルティングを行う HiDOF がスピンオフしている。また、Rodony Brooks 教授が 4 年間の秘密プロジェクトを経て発表した Rethink Robotics の Baxter ロボットも ROS がコアシステムとして活用されているという。前述のように ROS を使うことで様々な知能モジュールの活用と連携が容易になるため、これらの新興企業における ROS の活用は、今後ますます増えるだろう。

一方、従来型の産業用ロボットでも ROS を活用する動きがある。Yaskawa America と Southwest Research Institute が ROS-Industrial というプロジェクトを立ち上げ^{‡‡}、産業ロボットにおける ROS 応用を検討を始めた。また Enterprise ROS というユーザ主導のプロジェクト^{‡‡‡}では、企業が ROS を使った商品を開発する場合を想定した認証や長期サポートについて議論が始まっている。

3.3 ROS ツールの偏在化

本稿では最初に ROS の特徴としてツールとコミュニティの存在を指摘した。近い将来これらの特徴は、ロボティクス研究分野において当たり前の存在になり、ROS という存在は見えなくなると予想している。

例えばツール群は、ロボティクス研究にかかわらず有用であることから、ROS に依存しない独立したソフトウェアへと組み換えが行われている段階にある*。将来的には ROS のほとんどのツールは gcc や make/cmake と同様の一般のツールとして広く利用されるようになり、それが ROS と呼ばれるプロジェクトで開発されたものである、ということは、あえて指摘をしないと思ひ出さないようになるだろう。したがって数年後は ROS 以前と同様に、知能ロボットモジュールに焦点が当たる状況に戻るが、以前と違うのは、それが容易に開発できて、支持するコミュニティがあり、他のモジュールとの連携が容易になっているであろう。

[†]<http://www.ros.org/wiki/>, 約 1 万ページが存在し、毎日 50 件ぐらいが編集されている。

^{††}<http://code.ros.org/trac/>

^{†††}<http://answers.ros.org/>, 5000 件以上の質問が存在し、2000 人以上のユーザが参加している。

[‡]<http://www.ros.org/wiki/groovy/Planning>

^{‡‡}<http://www.ros.org/news/2012/01/ros-industrial-software-repository-now-available-from-swri.html>

^{‡‡‡}<http://www.ros.org/wiki/groovy/Planning/Enterprise>

*REP125 <http://ros.org/repos/rep-0125.html>,

REP114 <http://www.ros.org/repos/rep-0114.html> 等

図5 OpenRTM-ROSの相互運用シミュレーション。物理計算はOpenRTMが、視覚化はROSが行っている例

4. ROS との相互運用

本章ではROSのような勢いのあるプロジェクトに対してどのような対応をしたらよいか考えたい。新興企業や新しくロボット分野に参入する研究者にとってはROSを利用することで最先端の知能モジュールを活用し高いレベルから研究を開始できるためメリットしかないであろう。一方で、これまでも長期的に研究に取り組んできており、すでに様々な資産を蓄積している企業や研究室では過去の資産をすべて諦めて乗り換えるのか、あるいは、従前どおりの外部のプロジェクトとの連携を行わないで進めるのか判断は難しい。

そこで本章では「OpenRTM-ROS 相互運用プロジェクト」を紹介したい。これは、平成19年度から5年間実施した「次世代ロボット知能化技術開発プロジェクト」の一部として最終年度に実施されたものである。プロジェクト実施期間の間に海外でROSが急激に発展したため、知能化プロジェクトで採用しているOpenRTMミドルウェアとROSの間にはどのような連携の形があり、どう行うのが良いのか、その相互運用の在り方について明らかにすることが求められたという背景がある。

ここでは、「ROS = 通信ライブラリ + ツール + 基盤 + エコシステム」を参照し、ツールやエコシステムに関してはROSを利用しつつ、通信と基盤の共存を図った。これは、既存の通信ライブラリやソフトウェア基盤は活用しながらROSとの連携する方策であり、参考になれば幸いである。なお、ここで紹介するすべての内容は<http://code.google.com/p/rtm-ros-robotics/>で公開されている。

4.1 ROS ツールを活用した相互運用開発環境の構築

ROS ツールを利用し、他のパッケージと同様の感覚でOpenRTMが利用できるよう、`rosmake`をOpenRTMに対応した。具体的にはOpenRTMのインターフェース定義ファイルであるCORBAのIDLファイルの検出、コンパイル

図6 OpenRTM コンポーネントからROS ノードを自動生成している様子

ル、並びに生成されたインクルードファイルのパスの設定、ライブラリのパスとリンクの設定の機能を拡張した[†]。また、`roslaunch`をOpenRTMに対応させるためにコンポーネントの状態遷移、データポートおよびサービスポートの接続を行えるよう`rtconnect`, `rtactivate`という新しいタグを導入し拡張を行った^{††}。

これらの開発では、新規にOpenRTM ツールを開発せず、ROS ツールを拡張しOpenRTM 対応とすることで、省労力の開発と維持を可能にする戦略を採用した。

これにより、`rosmake hrpsys_ros_bridge`というコマンド一つで、`openrtm`, `openhyp3`, `hrpsys`の三つの知能化プロジェクトモジュールのインストールとコンパイルを、IDL ファイルのコンパイルとリンクも含めて行う。さらに、`roslaunch hrpsys_ros_bridge samplerobot_ros_bridge.launch`というコマンド一つで、図5のシミュレーションを起動するのに必要な10個以上のOpenRTMコンポーネントとROSノードを起動しOpenRTM, ROSのそれぞれの通信ポートを接続し、さらにOpenRTMのコンポーネントの活性化を行うことができるようになった。

4.2 OpenRTM-ROS 自動相互運用システム

ROSとOpenRTMの間で情報を変換するブリッジコンポーネントを自動的に生成することができれば、簡便にOpenRTMのコンポーネントをROSから利用できるだけでなく、将来の仕様変更にも明示的に対応する必要なく、継続的にブリッジコンポーネントを利用することができるようになる。

図6に開発したOpenRTM-ROS 自動相互運用システムを用いてOpenRTMのコンポーネントプログラムからROSのノードプログラムを生成する様子を示す。左図はOpenRTMによるヒューマノイドシミュレーションで、右図はそこで稼働しているOpenRTMコンポーネントの入出力インターフェース定義ファイルから自動生成したROSノードを示しており、ROSの入出力機能を利用してOpenRTMのヒューマノイドシミュレーション機能が利用できるようになっている。

5. おわりに

ここまで見てきたとおりROSの特徴は、通信や並列分

[†]<http://rtm-ros-robotics.googlecode.com/svn/tags/latest/rtmros-common/rtmbuild/cmake/rtmbuild.cmake>

^{††}<http://rtm-ros-robotics.googlecode.com/svn/tags/latest/rtmros-common/openrtm/scripts/rtmlaunch.py>

散といったソフトウェアのフレームワークよりも、開発者・利用者の立場に立ったツール群とコミュニティの存在にある。これにより、従来は管理が面倒だった大規模なロボットのソフトウェアを簡便で楽しくプログラムできるようになり、多くの開発者が喜んで ROS を使い、次々と新しい知能ロボットのソフトウェアが生まれてきている。また、開発されたソフトウェアはオープンにされているが、より重要なのはコミュニティが構築されていることにあり、その中でオープンになったソフトウェアが様々な人に様々な場面で活用され、問題点が洗い出され、報告され、フィードバックにつながり、ソフトウェアの高品質化を実現し、さらにより多くの開発者をひきつけるという、正のスパイラルが機能している。

また、ページ数の都合上、触れることができなかったが、ROS の開発を主導するにあたって PR2 ロボットの存在は無視できない。このロボットの特徴はソフトウェア開発者が使いやすい、という一点に尽きる。例えば緊急停止スイッチを押しても、もう一度押せば引き続き使えたり、柔らかなアームはどのような指令を送っても、ロボットも環境も破壊しない安心感がある。実際に WillowGarage の研究者は、このアームを足で蹴りながら利用していたが、そういう利用が可能なハードウェアがあるからこそ、開発された多くのソフトウェアが実ロボットでテスト・評価され、より高い品質に高められている。このような日本製のロボットがないのは大変残念である。

一方、我が国における ROS への取り組みは鈍いように見える。米国や欧州のように ROS を活用したプロジェクトも存在せず、日本製のほとんどのロボットは ROS をサポートしていない。robotics-worldwide のメーリングリストには ROS 関連の採用募集が 4 件/月のペースで流れているが、国内にはない。

一方で、ros.org/wiki サイトへのアクセスを国別に分けると米国をトップにドイツ、日本と続いており [1]、約 2,000 人程度のユーザがいることが推定[†]され、実は、かなりの規模で国内に ROS 開発者・利用者が存在することが分かる。プロジェクト推進側の方々がこの力を結集するような施策を立案することができれば強力な体制が構築できるだろうし、本稿を読んだ研究者・開発者の皆さんはひき続き、ロボットソフトウェアのコミュニティの活性化に寄与してほしい。初心者であれば、その基本的な疑問こそがソフトウェアをより使いやすいものにするので大変貴重な情報になっているし、すでに使い始めていれば数多くのバグ報告や機能提案することで、将来的には自分を楽しめるだろう。また、海外を中心としたプロジェクトであり英語が主に利用

されているが、母国語で堂々と質問し、別の人が英語に翻訳している例もある。さらに、コミュニティサイトは誰もが加筆・修正できるようになっているため、読みづらい英語は気がついた人が直してくれる。本稿がオープンなロボットソフトウェアコミュニティへの積極的な参加や支援への契機になれば幸いである。

参考文献

- [1] Tully Foote. ROS Community Metrics, <http://pr.willowgarage.com/downloads/metrics/metrics-report-2012-07.pdf>, June 2012.
- [2] A crowd-sourced map of ROS users around the world, <http://maps.google.com/maps/ms?ie=UTF&mmsa=0&msid=209668390659853657363.00049c608b78bc7779683>, 2012.
- [3] M. Saito, H. Chen, K. Okada, M. Inaba, L. Kunze and M. Beetz: "Semantic object search in large-scale indoor environments," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World, pp.25–30, 2011.
- [4] B. Gerkey: What is ROS exactly? Middleware, Framework, Operating System?, <http://answers.ros.org/question/12230/what-is-ros-exactly-middleware-framework-operating>, December 2011.
- [5] M. Quigley, E. Berger and A.Y. Ng: "Stair: Hardware and software architecture," AAAI 2007 Robotics Workshop, pp.31–37, 2007.
- [6] M. Quigley, K. Conley, B.P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A.Y. Ng: "ROS: an open-source Robot Operating System," ICRA Workshop on Open Source Software, 2009.
- [7] B. Gerkey: Open Source Robotics, November 2008. <http://www.willowgarage.com/blog/2008/11/17/steve-cousins-speaking-robo-development-tuesday>.
- [8] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J.M.M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle and R. van de Molengraft: "Roboearth," Robotics Automation Magazine, vol.18, no.2, pp.69–82, 2011.
- [9] E.S. Raymond: The Cathedral and the Bazaar, <http://www.catb.org/~esr/writings/cathedral-bazaar/>, 日本語訳 山形浩生: 伽藍とバザール, <http://cruel.org/freeware/cathedral.html>.

岡田 慧 (Kei Okada)

1997 年京都大学工学部情報工学科卒業。2002 年東京大学大学院工学系研究科情報工学専攻博士課程修了。博士 (工学)。2002 年東京大学大学院情報理工学系研究科科学技術振興特任教員。2006 年同特任講師。2009 年同大学准教授となり現在に至る。ヒューマノイド統合システム、実時間三次元ビジョンの研究に従事。1999 年第 4 回ロボティクスシンポジウム優秀論文賞、2006 年 IEEE-RAS International Conference on Humanoid Robots Best Paper Award 受賞。情報処理学会、IEEE の各会員。

(日本ロボット学会正会員)

[†]アクセス (visit) が 136,298 でユーザ (unique visitors) が 40,166。日本からの visit は 6,725 から単純な比例計算を行った。