研究ノート

ROSを用いたロボットシステムの構築

平出貴大[*1]、木村宏樹[*1]、依田康宏[*1]

Construction of Robot System Using ROS

Takahiro HIRADE[*1], Hiroki KIMURA[*1] and Yasuhiro YODA[*1]

Industrial Research Center[*1]

オープンソースであり豊富なツールやライブラリを持つロボット開発用のソフトウェアプラットフォーム ROS(Robot Operating System)を用いて、生産現場の作業工程の自動化を模擬したロボットシステムを構築した。ROS により、アームロボットの動作制御やセンサ、ベルトコンベア等の周辺機器との連携制御が可能であることを確認し、ワークの仕分け作業を自動化するアプリケーションを作成した。

1. はじめに

近年、少子高齢化の進行に伴い生産年齢人口が減少し、それに起因する深刻な労働力不足が社会課題となっている。このような状況下において、各産業分野では省人化や業務効率化を目的とした自動化への関心が高まっており、特にロボット技術の活用が強く期待されている。一方、ロボットを活用した自動化の実現には、高度なプログラミング能力やシステムインテグレーション技術が求められる場合が多く、特に技術的リソースに乏しい中小企業にとってはその活用が容易ではないという課題がある。このような課題に対し、汎用的で再利用性を考慮したシステム構築の方法として、オープンソースソフトウェアの利用が進められている[1)]。

そこで本研究では、自動化を検討する企業への技術支援に活用することを目的に、オープンソースで豊富なツールやライブラリを持つソフトウェアプラットフォームROS(Robot Operating System)[2)]を用いてロボットシステムの構築を試みた。

2. 実験方法

2.1 システムの構想設計

自動化する作業工程として、ロボットがベルトコンベアを流れる2種類のワークを仕分けるシステムを構築することとした。ワークを把持して移動させるロボットには(株)デンソーウェーブ製のアームロボット「COBOTTA」を用い、ワークの検知には光電センサを、ワークの種類判別にはカメラとマイコンを用いた。

2.2 ROS 環境の構築

PC でアームロボットやベルトコンベア等を制御するアプリケーションを作成するため、PCにROSの開発環境を構築した。ROS 環境の構築にはアプリケーションをコンテナという単位でパッケージ化して実行・管理するプラットフォーム「Docker」[3)]を用い、ROS の公式コンテナ[4)]をもとに構築した。作成したアプリケーションを新たにコンテナとしてパッケージ化することで、OS やバージョンが異なる環境でも Docker 上で動作可能となり、高度な技術を必要とせずに同じアプリケーションを実行するシステムを構築することができる。

2.3 ROSによるアームロボットの動作制御

アームロボットの動作プログラミングは(株)デンソーウェーブと(一社)東京オープンソースロボティクス協会が ROS 用に公開しているプログラム[5),6)]をもとに行った。ロボットメーカー各社でプログラミングのコマンドが異なるが ROS では共通化されているため、位置と姿勢、グリッパの開閉の情報を入力するだけで所望の動作をプログラミングすることができた。

作成した動作プログラムをアームロボットに実装する前に確認するため、ROS と連携可能なツールである動力学シミュレータ「Gazebo」[7)]を用いてシミュレーション空間を作成した(**図 1**)。アームロボットの 3D モデ

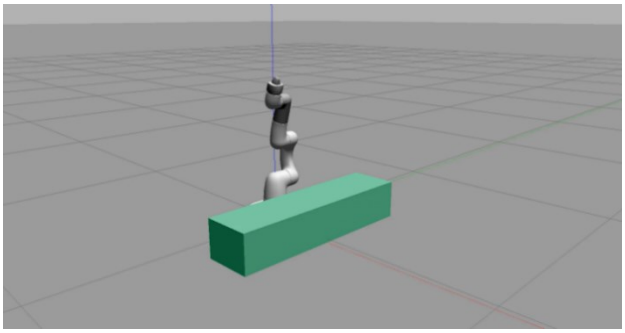

図 1　構築したシミュレーション空間

[*1] 産業技術センター　自動車･機械技術室

ルと、ベルトコンベアと同サイズの箱型構造体を配置することで、アームロボットとベルトコンベアが接触しないことを確かめた。

2.4 ROS による周辺機器との連携制御

センサ等の周辺機器を含めたシステム構成を**図 2** に示す。アームロボットを PC とイーサネット接続し、ベルトコンベア、光電センサ、マイコンはアームロボットの I/O ポートに接続した。光電センサとマイコンからの入力信号に応じてアームロボットとベルトコンベアへの出力信号を制御することで、連携制御するアプリケーションを作成した。

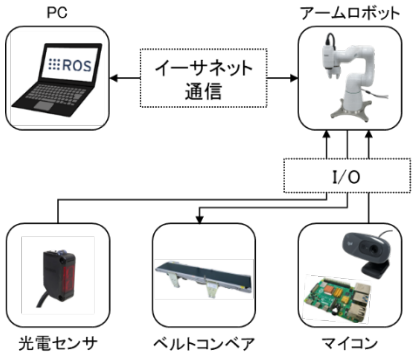

図 2　システム構成図

2.5 ロボットシステムの構築

構築したロボットシステムの外観図を**図 3** に、アームロボットによるワークの仕分け作業の動作順序を**表 1** に示す。表 1 の動作はシステムを起動してから停止するまで繰り返し行う。

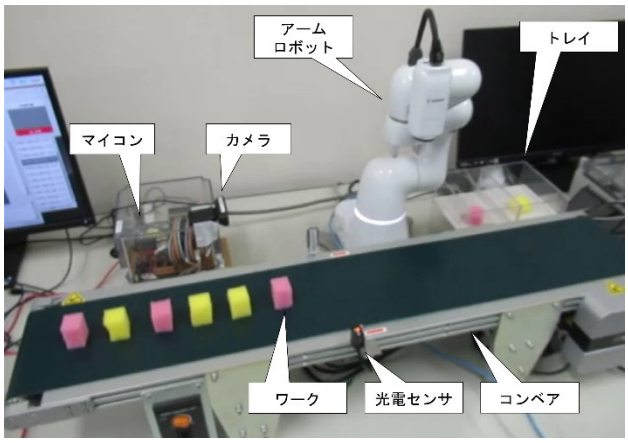

図 3　ロボットシステムの外観図

3. 実験結果及び考察

構築したロボットシステムの動作を確認するための実験を行った。図 3 に示すように色違いの 2 種類のワークを各々のトレイに仕分けることができることを確認した。なお、事前にシミュレーション空間上で動作検証した結果、アームロボットが動作中にベルトコンベアと接触しないことが確認できた。また、今回作成したアプリケーションは、Docker コンテナとしてパッケージ化することで他の環境でも容易に実行可能である。

表 1　ロボットシステムの動作順序

順序	動作
1	ベルトコンベアが駆動してワークが移動する
2	ワークが光電センサの光を遮りベルトコンベアが停止する
3	アームロボットが駆動してワークを把持してカメラの前に移動する
4	カメラ画像からマイコンがワークの種類を判別する
5	判別結果をもとにアームロボットがワークを所定のトレイに置く
6	アームロボットが初期位置に戻りベルトコンベアが再び駆動する

4. 結び

本研究の結果は以下のとおりである。

(1) ROS を作業工程の自動化を模擬したロボットシステムを構築した

(2) ROS の開発環境を構築して作成したアプリケーションにより、アームロボットの動作制御や周辺機器との連携制御が可能であることを確認した。

(3) アームロボットの接触確認など動作プログラミングの効率化をするためのシミュレーション空間を構築した。

文献

1) (地独)大阪産業技術研究所: ROS を用いたシステム構築技術 3 ROS2 を用いた卓上自動化システムの構築, https://orist.jp/technicalsheet/23-12.pdf, (2025/5/14)

2) Open Robotics: ROS - Robot Operating System, https://www.ros.org/, (2025/5/14)

3) Docker Inc.: Accelerated Container Application Development, https://www.docker.com/,(2025/5/14)

4) Open Robotics: Open Robotics. GitHub, https://github.com/osrf, (2025/5/14)

5) (株)デンソーウェーブ: DENSORobot. GitHub, https://github.com/DENSORobot, (2025/5/14)

6) (一社)東京オープンソースロボティクス協会: Tokyo Opensource Robotics Kyokai Association. GitHub, https://github.com/tork-a, (2025/5/14)

7) Open Robotics: Simulate before you build, https://gazebosim.org/home, (2025/5/14)