RTM と ROS の共通座標系管理システム

操作マニュアル

名城大学メカトロニクス工学科

ロボットシステムデザイン研究室

2023 年 12 月 14 日

目次

1. はじめに

1.1. 目的

異種ミドルウェア相互運用のための座標変換コンポーネントの開発において，座標変換コンポーネントの開発を行った．また，座標変換コンポーネントの実装事例として RTM と ROS の共通座標系管理システムの開発を行った．本書は RTM と ROS の共通座標系管理システムの使い方を説明することを目的とする．

1.2. 本書を読むにあたって

本書は RT ミドルウェアに関する基礎知識を有した利用者を対象としている．また， 各 RTC の詳細な仕様等については同ファイル内の仕様解説マニュアルを参考にすること．

1.3. 動作環境

本 RTC の動作確認環境を以下に示す．

OS	Ubuntu 20.04
RT ミドルウェア	OpenRTM-aist-2.0
ROS	ROS noetic

1.4. 開発環境

本 RTC の開発環境を以下に示す．

OS	Ubuntu 20.04
言語	Python
RT ミドルウェア	OpenRTM-aist-2.0

2. コンポーネントの準備

2.1. ROS の既存コンポーネント

今回のシステムでは MikataArm を利用したマニピュレーションを行った．以下のサイトを参考にして環境構築を行うこと．

https://github.com/Ryusei-Tomikawa/DYNAMIXEL-MikataArm-

2.2. OpenRTM の既存コンポーネント

今回のシステムで再利用した OpenRTM のコンポーネントを示す．

- SingleArUco2

  https://github.com/rsdlab/SingleArUco2

- WebCamera2

  https://github.com/rsdlab/WebCamera2

- ImageViewer2

  https://github.com/rsdlab/ImageViewer2

これらと今回作成したコンポーネントを用いてシステム構築を行った．システム構築については以降で説明する．

2.3. ROS の新規作成コンポーネントの準備

新規作成コンポーネントである MikataArm_manipulation.py をシステムに用いる．以下のコマンドで環境構築を行う．

```
$ git clone https://github.com/itsuku-kito/RTM-management-of-coordinate-systems
$ cd RTM-management-of-coordinate-systems
$ mv ROS(RTM-management-of-coordinate-systems) catkin workspace
$ cd ..
$ catkin build
```

2.4. OpenRTM の新規作成コンポーネントの準備

新規作成コンポーネントである Coordinate_transformation_RTC と Coordinate_transformation_RTC_typecasting をシステムに用いる．2.3ROS の新規コンポーネントの準備の際に gitclone したファイルの中の RTC ファイルを OpenRTM のコンポーネントを入れるディレクトリ内に入れる Coordinate_transformation_RTC_typecasting は SingleArUco2 内で使われてい

る独自のデータ型である arUcoMakerPose3D に Coordinate_transformation_RTC を対応させるためのコンポーネントである．

3. システム概要

今回作成したシステムは，「2.コンポーネントの準備」部分に記載のコンポーネントを用いた OpenRTM と ROS を相互運用したマニピュレーションシステムである．今回は WebCamera2 のコンポーネントではロジクール BRIO 4 K ウェブカメラを動かし，マニピュレータとして MikataArm を動かした．

コンポーネントがどのミドルウェアで作成されたかを明示したシステム構成図を図 1 に示す．また，動作の概要を図 2 に示す．

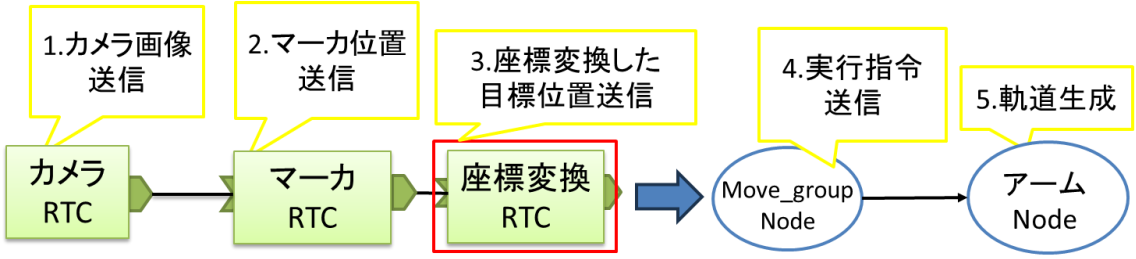

図 1 システム構成図

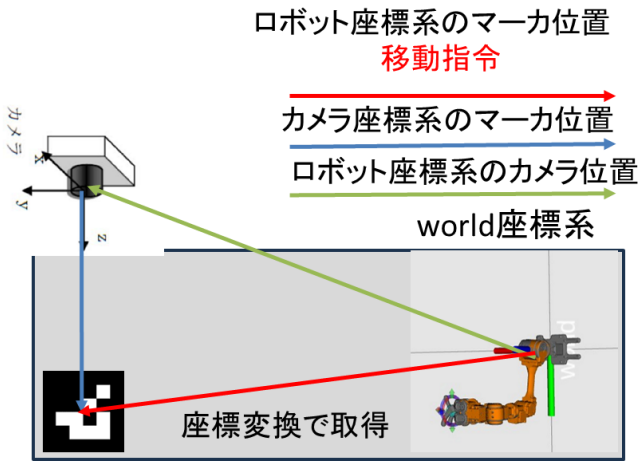

図 2 動作概要

今回作成したコンポーネントは赤枠で囲われたものである．カメラ RTC で取得するカメラ画像をマーカ RTC に送信し，マーカ RTC で取得するマーカ位置を座標変換 RTC に送信し，座標変換するまでを OpenRTM で行い，座標変換して得た目標位置を ROStransport で送信する．その後，目標位置を受け取ってアームが動き出すシステムになっている．

今回開発した座標変換コンポーネントを用いることで既存のコンポーネントを作り換える必要なく座標変換を行うことができた.

4. システムの操作方法

4.1. roscore の起動

ターミナルにて roscore を実行する.

```
~$ roscore
```

4.2. RTC の起動

RTC を立ち上げて図 1 のように接続する.

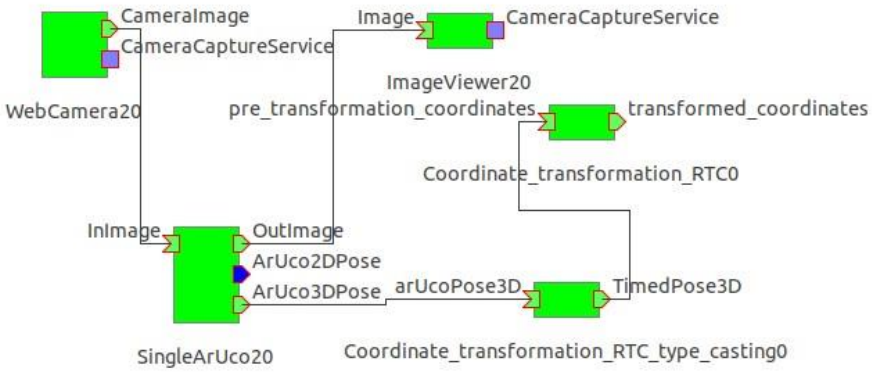

図 3　RTC 接続図

Coordinate_transformation_RTC の起動の際には以下のコマンドで起動することに注意する.

```
$ python3 Coordinate_transformation_RTC-f rtc.conf
```

4.3. スクリプトの実行

まず，https://github.com/Ryusei-Tomikawa/DYNAMIXEL-MikataArm-の実機環境で行う場合のコマンドである以下のコマンドを実行する

```
$ sudo chmod 666 /dev/ttyUSB0
$roslaunch open_manipulator_controller open_manipulator_controller.launch
```

```
use_moveit:=true
```

USB の実行権限に関しての Path は適宜変更することその後，新規ターミナルまたは新規タブを開き，以下のコマンドを実行する．

```
~$ rosrun Coordinate_transformation_ROS MikataArm_manipulation.py
```

以上でシステム構築は終了である．