

Development and Visualization of a Dual-Arm Robotic System in ROS

**AUT.841-2024-2025-1 Robot Manipulators: Modeling, Control and Programming**

**Mikhail Silaev**

**Juha Jalkanen**

**Amirmohammad Daareyni**

Table of contents

[introduction 3](#_Toc182410139)

[Environment Setup 3](#_Toc182410140)

[Step 1: Create Workspace 3](#_Toc182410141)

[Step 2: Download and Configure Motoman Package 3](#_Toc182410142)

[Step 3: Create Project Packages 4](#_Toc182410143)

[Step 4: Build and Launch the Project 5](#_Toc182410144)

[Step 5: Launch Dual-Arm Robot with MoveIt Configuration 5](#_Toc182410145)

[Dual-Arm Robot Configuration 6](#_Toc182410146)

[Positioning 6](#_Toc182410147)

[MoveIt Configuration 7](#_Toc182410148)

[Testing and Verification 8](#_Toc182410149)

[ROS Nodes and Topics 9](#_Toc182410150)

[rqt\_graph Overview 9](#_Toc182410151)

# introduction

This report presents the development of a dual-arm robotic system using two SIA20D Motoman robots equipped with WSG50 parallel grippers. The primary objective is to create a modular URDF configuration for the dual-arm setup and integrate it with MoveIt for motion planning. The entire system is visualized in RVIZ, allowing users to control and test individual motion groups, including each arm, gripper, and the dual-arm combination. The report details the implementation steps, configuration process, and initial testing results, laying the groundwork for further application in pick-and-place tasks.

# Environment Setup

In this section, we outline the steps taken to set up the workspace and integrate the required robot and gripper packages for the dual-arm robotic system.

### Step 1: Create Workspace

To begin, we created a dedicated workspace for this assignment which is called assignment\_ws using following commands:

mkdir assignment\_ws

cd assignment\_ws

### Step 2: Download and Configure Motoman Package

We cloned the Motoman package from the ROS-Industrial repository, specifically using the kinetic-devel branch to ensure compatibility:

git clone -b kinetic-devel https://github.com/ros-industrial/motoman.git src/motoman

Next, we updated the dependencies and installed any missing ROS packages:

rosdep update

rosdep install --from-paths src/ --ignore-src --rosdistro noetic

We then built the workspace and sourced the setup file:

catkin\_make

source devel/setup.bash

To verify the installation, we launched the Motoman SIA20D MoveIt configuration in RVIZ:

roslaunch motoman\_sia20d\_moveit\_config demo.launch

If everything is setup properly, you will see a robot in RVIZ similar to Figure 1.

A screenshot of a computer

Description automatically generated

Figure 1 SIA20D Robot Initialized in RVIZ with MoveIt Interface

### Step 3: Create Project Packages

To modularize the dual-arm setup, we created two packages:

1. **my\_sia20d**: This package contains the URDF and configuration files for the SIA20D Dual arm Motoman robot.
2. **wsg50\_gripper**: This package includes the URDF configuration for the WSG50 parallel gripper.

Commands to create the packages:

cd src

catkin\_create\_pkg my\_sia20d urdf

catkin\_create\_pkg wsg50\_gripper urdf

The necessary files from the provided folders (my\_sia20d and wsg50\_gripper) were then copied into these newly created packages.

### Step 4: Build and Launch the Project

We built the updated workspace with the added packages:

cd ..

catkin\_make

source devel/setup.bash

To test the integration, we launched my\_sia20d package:

roslaunch my\_sia20d test\_sia20d.launch

Figure 2 indicates the dual arm robot in RVIZ.

A screenshot of a computer

Description automatically generated

Figure 2 Dual arm robot

### Step 5: Launch Dual-Arm Robot with MoveIt Configuration

Finally, we launched the MoveIt configuration for the assignment to visualize and control the dual-arm robotic system:

cd ~/assignment\_ws

source devel/setup.bash

roslaunch my\_moveit2 demo.launch

This setup provided the foundation needed for configuring the motion groups and visualizing the dual-arm robot in RVIZ. Figure 3 represents the dual arm robot in home (a) as well as a random position (b).

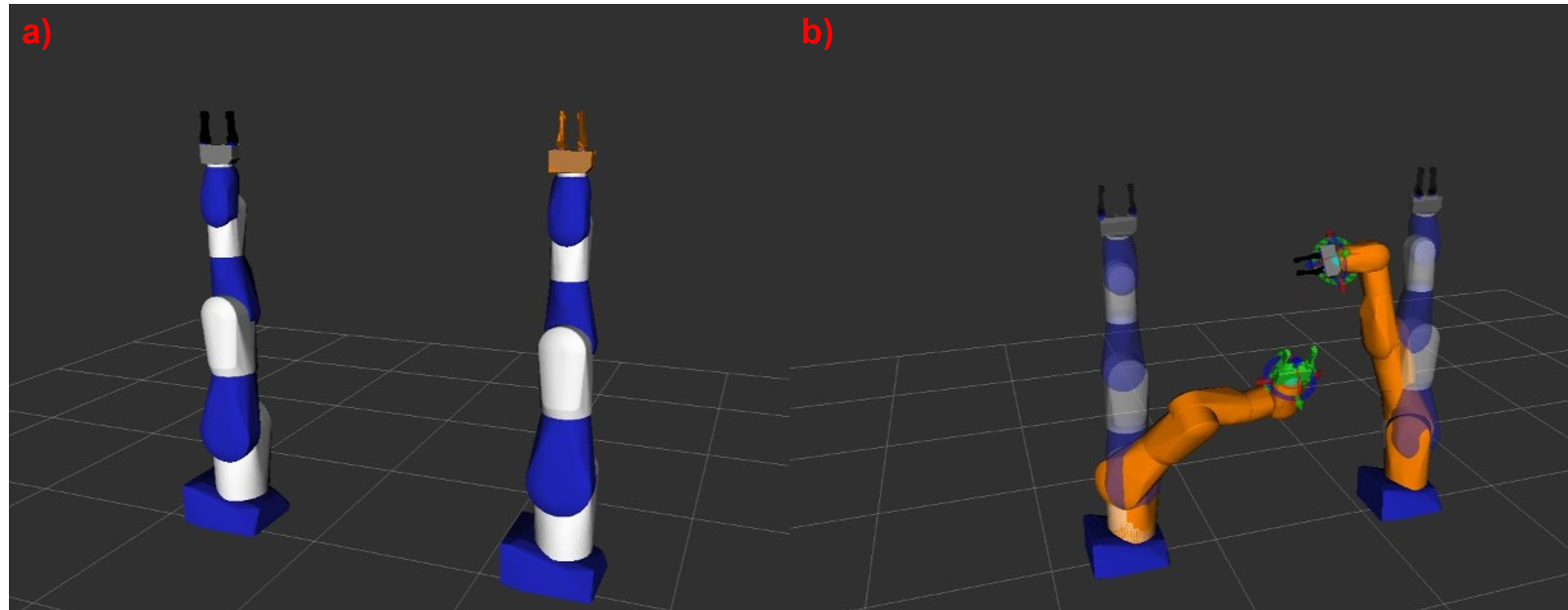


Figure 3 Dual arm robot with gripper in RVIZ. a) Home position, b) random position.

# Dual-Arm Robot Configuration

### Positioning

#### Robot Positioning Relative to the World Frame:

* In the dual-arm setup, precise positioning was required to enable effective coordination between the robots. Robot 1 is positioned at coordinates (0.7, 0, 0.7) meters, and Robot 2 at (-0.7, 0, 0.7) meters relative to the world frame. This symmetry along the x-axis allows both robots to operate within a shared workspace and reach common objects, facilitating pick-and-place operations.
* **Static Transform Publisher**: A static transform between the world frame and each robot’s base link (one\_base\_link) was defined to anchor each robot’s position relative to the workspace. This transform is broadcasted using the tf2\_ros package, ensuring the robots remain aligned within the MoveIt planning environment.

### MoveIt Configuration

##### Configuration Package Creation:

* A MoveIt configuration package was created to support the dual-arm system, with settings for motion planning, kinematics, and collision avoidance. This package enables real-time visualization and interaction through the RViz environment.
* **Planning Pipeline**: The OMPL (Open Motion Planning Library) pipeline was selected as the default for generating motion plans due to its efficient algorithms suited for multi-DOF robotic arms.

#### Defining Motion Groups:

* **Left Arm (Robot 1)**:
  + A motion group for Robot 1’s arm was defined to handle all degrees of freedom specific to that robot. This group allows Robot 1 to independently reach different positions in the workspace.
* **Right Arm (Robot 2)**:
  + Similarly, the Right Arm motion group manages the movements of Robot 2, ensuring that it can operate autonomously from Robot 1 while remaining within a safe range.
* **Left Gripper (Robot 1 Gripper)**:
  + The Left Gripper motion group controls the WSG50 gripper attached to Robot 1, enabling precise object gripping and release functions for pick-and-place tasks.
* **Right Gripper (Robot 2 Gripper)**:
  + The Right Gripper motion group operates the WSG50 gripper on Robot 2, complementing the Right Arm’s reach and grip capability.
* **Dual-Arm Motion Group**:
  + This motion group combines both Left and Right Arm motion groups, enabling synchronized movements. This group is essential for dual-arm tasks that require both robots to coordinate, such as transferring objects from one arm to the other or jointly moving larger objects.
  + **Collision Avoidance and Synchronization**: The dual-arm motion group setup ensures that both arms can move without interfering with each other, using built-in collision detection within MoveIt. This was achieved by specifying joint constraints and defining allowed planning spaces for each arm.

#### Execution Mode:

* **Fake Controllers**: Given the absence of physical hardware, fake controllers were set up to simulate robot movements, with the execution mode configured as interpolate. This setting ensures smooth, interpolated joint movements, making the simulated actions visually similar to actual robot behavior.
* **Joint State Publisher**: A joint\_state\_publisher was used to broadcast fake joint states, enabling real-time updates in RViz without requiring actual joint data from hardware.

### Testing and Verification

##### RViz Visualization and Motion Testing:

* The MoveIt launch file included settings to load the robot model into RViz, providing an interactive visualization of both robots and grippers. Each motion group was tested in RViz to verify that:
  + **Individual Arm Movements**: Both Left and Right Arm motion groups could move independently within their respective ranges.
  + **Gripper Operations**: The Left and Right Grippers successfully opened and closed, with accurate visualization of their positions.
  + **Dual-Arm Coordination**: The Dual-Arm motion group demonstrated synchronized movement, with both robots operating smoothly and avoiding self-collision. This test confirmed that the system could perform coordinated actions, such as transferring objects between the two arms.
* **Video Demonstration**: A video recording captures each of these movement tests, providing a visual overview of the Arms, grippers, and Dual-Arm motion groups in action. The video demonstrates the system’s capability for both independent and coordinated operations, showcasing the range, precision, and interaction of both robots.

# ROS Nodes and Topics

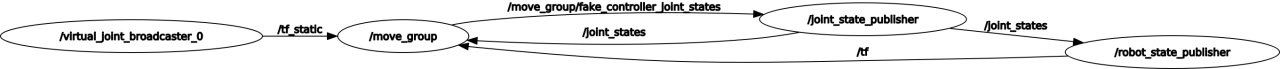


Figure 4 rqt\_graph visualization of the ROS nodes and topics in the dual-arm robotic system.

### rqt\_graph Overview

#### Graphical Representation:

* The rqt\_graph screenshot (see Figure 4) shows the active nodes and topics involved in the dual-arm setup, highlighting the communication structure necessary for the operation of each robot arm and gripper. This visualization helps to understand how information flows between nodes and the role of each topic in managing robot states and transformations.

#### Key Nodes and Their Functions:

* **/move\_group**: This is the central node for motion planning, provided by MoveIt. It handles the calculation of motion plans and manages the control of the robot's arms and grippers. It interacts with other nodes to send joint state commands and update the robot’s positions.
* **/joint\_state\_publisher**:
  + This node publishes the simulated joint states for the robot’s arms. It provides the necessary joint data for real-time visualization and interaction within RViz.
* **/robot\_state\_publisher**: The robot\_state\_publisher node calculates and broadcasts transformations (tf) for each robot link based on joint states. It ensures that the robot’s kinematic chain is represented accurately within the world frame.
* **/virtual\_joint\_broadcaster\_0**: This node publishes a static transformation between the world frame and the base of each robot. It’s crucial for anchoring the robots within the shared workspace, allowing accurate planning and visualization.

#### Key Topics and Their Roles:

* **/joint\_states**: This topic is published by the /joint\_state\_publisher and consumed by /robot\_state\_publisher. It contains real-time updates of each joint’s position, enabling RViz to display the robot’s movements.
* **/tf** and **/tf\_static**: These topics broadcast the transformation tree, which establishes the spatial relationships between various links of the robot. /tf is used for dynamic transformations (joint movements), while /tf\_static maintains the fixed transform between the world and the robot base link.