MANCHESTER 1824
The University of Manchester

Research on remote control methods for a 7- DOF manipulator

A dissertation submitted to The University of Manchester for the degree of
Master of Science in Robotics
in the Faculty of Science and Engineering

Year of submission
2024

Student ID
11428365

School of Engineering

Contents

Contents . . . 2

Terms and abbreviations . . . 4

Abstract . . . 5

Declaration of originality . . . 6

Intellectual property statement . . . 7

Acknowledgements . . . 8

1 Introduction . . . 9
1.1 Background and motivation . . . 9
1.2 Terminology and explanations . . . 9

2 Literature review and previous study . . . 11
2.1 Introduction to Robotic Manipulators . . . 11
2.2 Teleoperation Challenges and Solutions . . . 12
2.3 Simulation and Modeling in Robotics . . . 12
2.4 Conclusion and Gaps in Current Research . . . 12

3 Methods . . . 13
3.1 URDF Wireless Transmission and Display . . . 13
3.2 FR3 manipulator modelling and simulation . . . 13
3.3 Real-time status monitoring and visualization . . . 14
3.4 Remote Control Implementation . . . 14
3.5 Motion Planning and Control . . . 14

4 Results and discussion . . . 15
4.1 Implementation and effect of URDF wireless transmission . . . 15
4.2 Simulation and control of FR3 manipulators . . . 16
4.3 Remote control of the manipulator . . . 19

5 Conclusions and outlook . . . 22
5.1 Conclusion . . . 22
5.2 Outlook . . . 23

References . . . 24

Appendices . . . 29

A Project outline . . . 29

B Risk assessment . . . 30

C Work on github . . . 37

D TF tree **37**

E Python control code **37**

Word count: 4390

Terms and abbreviations

FR3 **F**ranka **R**esearch **3**

ROS **R**obot **O**perating **S**ystem

URDF **U**nified **R**obot **D**escription **F**ormat

xacro **X**ML m**acros**

XML **Ex**tensible **M**arkup **L**anguage

RViz **R**econstruct **Vi**sual**iz**ation Interface

FCI **F**ranka **C**ontrol **I**nterface

API **A**pplication **P**rogramming **I**nterface

VR **V**irtual Reality

AR **A**ugmented **R**eality

Abstract

This project explores the simulation and remote control of the FR3 robotic arm, focusing on wireless URDF transmission, simulation, and real-time control of the physical robot. We successfully implemented wireless transmission of URDF files using ROS, enabling efficient sharing and visualization of the robot model across devices. In the simulation phase, the FR3 robotic arm was modeled and controlled through a combination of Gazebo, RViz, and MoveIt, with additional Python scripting for precise end-effector control. Finally, the real FR3 robotic arm was integrated into the system for remote control via a two-device setup, demonstrating stable performance. Future work will aim to enhance system stability, monitoring capabilities, and data visualization to improve the overall robustness and applicability of the system in various industrial and research scenarios.

Keywords: FR3 robotic arm, ROS, URDF transmission, simulation, remote control, Gazebo, RViz, MoveIt, Python scripting, end-effector control, system stability.

Declaration of originality

I hereby confirm that this dissertation is my own original work unless referenced clearly to the contrary, and that no portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Intellectual property statement

i The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made *only* in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see `http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420`), in any relevant Dissertation restriction declarations deposited in the University Library, and The University Library's regulations (see `http://www.library.manchester.ac.uk/about/regulations/_files/Library-regulations.pdf`).

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my parents for raising me and supporting my decision to study in the UK. Without their unwavering love and encouragement, none of this would have been possible.

I would also like to extend my heartfelt thanks to all of my teachers at the University of Manchester who have guided and supported me throughout my academic journey. Special thanks to Dr. Simon Watson, Dr. Andrew West, Dr. Kieran Wood, Dr. Pawel Ladosz, Dr. Mingfei Sun, Dr. Angelo Cangelosi, Dr. Murilo Marinho, and Dr. Xiaoxiao Cheng for their invaluable teaching and mentorship.

A special note of appreciation goes to my project supervisor, Dr. Wei Pan, and my senior, Yiyu Jiang, for their helpful guidance and advice during this project. Their insights and support were instrumental in the successful completion of this work.

I would also like to thank my groupmates, Sen Lu, Nathon Shankar, and Mahmod El Shal, for their collaboration and teamwork. Additionally, I am grateful to my fellow Robotics classmates—Hangzheng An, Jingchuan Lin, Di Liu, Zhaocheng Li, Jiahao Geng, Yining Wang, and Xingjian Zhang—for always being there to lend a helping hand when I needed it.

Finally, I want to express my deep appreciation to the friends I made in Manchester—Wenjie Zhai, Yubo Yan, Zhiyu Xu, Hangjia Hu, Weixiang Lin, Yuzhen Niu, Shuangqiao Xu, Zixiao Peng, Zhiyan Qiao, and many others whose real names I may not know. Your emotional support and friendship made this year a joyful and memorable experience. Thank you all for being a part of my journey.

1 Introduction

1.1 Background and motivation

With the rapid advancement of automation and robotics, manipulators have become essential in various industries, including manufacturing, medical, and service sectors, excelling in high precision, repetitive tasks that boost productivity and maintain quality[1]. In the medical field, they enhance precision in delicate surgeries, reducing human error and shortening operation times[2].

This project focuses on the Franka Research 3 (FR3)[3] manipulator, shown as Fig.1 from Franka Emika, a cutting-edge robotic system known for its high precision and flexibility. Capable of performing a wide range of complex tasks across research and industrial applications, the FR3's potential is often limited by traditional control methods that rely on local control, thus restricting its effectiveness in teleoperation and multi-site collaborations.

Fig. 1. Franka Research 3 manipulator model[4]

As industries and research environments become more interconnected, remote control of manipulators is becoming essential for applications like distributed manufacturing, remote maintenance, and teleoperated medical procedures[5]. This capability enhances flexibility, reduces the need for direct human intervention, and improves safety and efficiency.

This project tackles the challenge of developing a remote control solution for the FR3 manipulator by modeling and simulating the system in a virtual environment. This approach allows for thorough testing and optimization of control algorithms, reducing risks and costs before real-world deployment. The goal is to establish a strong foundation for future FR3 remote operations and explore its potential to transform various industries with increased operational versatility.

1.2 Terminology and explanations

Accurate understanding of key terms and concepts is crucial for successful robot simulation and remote control projects[6]. This chapter explains the terms, technical concepts, and principles related

to FR3 manipulator simulation and remote control. Clear definitions will help articulate technical details and enhance the reader's understanding of the project's context and objectives, laying a solid foundation for the discussions in the following chapters[7].

1. FR3

   The FR3, developed by Franka Emika, is a versatile robotic manipulator known for its adaptability, high programmability, and ease of integration. Equipped with force feedback technology, it excels in precise tasks across industrial automation, research, and medical fields. Its modular design and efficient control system make it an ideal platform for diverse applications.

2. ROS

   ROS (Robot Operating System) is a popular open-source framework for robotics development, providing tools, libraries, and standard interfaces that simplify the design and deployment of robotic systems[8]. It supports multiple programming languages and a modular architecture, enabling seamless integration of components like sensing, control, navigation, and communication. Thanks to its flexibility and strong community support, ROS is widely used in both academic research and industry, allowing developers to efficiently build complex applications and easily switch between hardware platforms and simulation environments.

Fig. 2. The ROS Ecosystem[8]

3. URDF and xacro

   URDF (Unified Robot Description Format) and xacro (XML Macros) are key tools in ROS for defining and managing robot models[9]. URDF describes a robot's physical structure, including links, joints, mass, and inertia, for both simulation and real-world use, forming the basis for robot simulation and control. As Fig.3 shows, xacro extends URDF by allowing the creation of URDF files through reusable macro modules, simplifying complex models, reducing redundancy, and boosting development efficiency. Together, URDF and xacro streamline robot modeling, improving maintainability and extensibility.

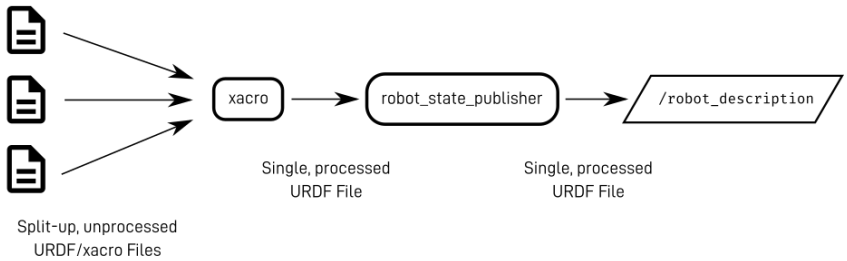

Fig. 3. Describing robots with URDF[9]

4. RViz and Gazebo

RViz (ros-visualization) is a versatile ROS tool for visualizing and debugging robot states, sensor data, and environmental information[10]. It allows users to view various aspects of a robotic system, such as position, orientation, path planning, LIDAR scans, and camera images, helping developers troubleshoot and refine their systems. RViz's flexibility with plugins and display types makes it an essential tool for robot development.

Gazebo, often used with ROS, is a powerful simulation tool that models complex 3D worlds with realistic physical properties, collision detection, and sensor data generation[11]. It enables developers to test and validate robotic systems in a virtual environment, reducing the reliance on real hardware and lowering development costs and risks[12]. Gazebo's integration with ROS allows for a seamless transition from simulation to real-world robotic applications, shown in Fig.4.

Fig. 4. The integration between ROS-Gazebo and RViz[13]

5. MoveIt

   MoveIt is a powerful motion planning framework for ROS robots, which is widely used for robot arm motion control, path planning, collision detection, inverse kinematics, operation execution, and other tasks[14]. With MoveIt, developers can implement anything from simple point-to-point motion to complex multi-degree-of-freedom path planning, ensuring that the robot can avoid obstacles and reach its target position while performing its tasks[15]. MoveIt supports a wide range of control interfaces and hardware platforms, enabling it to be integrated and used with real robots and simulation environments[16]. It also provides user-friendly visualization tools and plug-ins to help users intuitively debug and optimize robot motion, making it a key tool in industrial automation, robotics research, and application development[17].

2 Literature review and previous study

2.1 Introduction to Robotic Manipulators

With the rapid advancements in automation and robotics, robotic manipulators have become crucial in various industries[18]. In industrial settings, these manipulators enhance productivity and product quality through their precision and ability to perform repetitive tasks[19]. For example, in assembly, welding, and packaging, they reduce manual operation uncertainties by maintaining consistent accuracy[20][21]. Beyond industry, manipulators also show great promise in the medical field, particularly in minimally invasive surgery, where they perform delicate operations that reduce surgical risks and recovery times[22].

2.2 Teleoperation Challenges and Solutions

As technology advances, teleoperation is becoming a key research area in robotics[23][24]. It allows operators to control robots remotely, crucial for applications in telemedicine and hazardous environments[25][26]. Teleoperation depends on real-time control and feedback, with progress in network communications, sensor fusion, and human-machine interfaces playing a critical role[27].

Fig. 5. Example of manipulator remote control[28]

Teleoperation faces challenges like network latency and packet loss, critical in fast-response applications such as telesurgery [29], [30]. Latency delays robot responses, compromising accuracy and safety [30], while packet loss can lead to miscommunication and potential dangers [31], [32]. To address these, researchers have developed efficient communication protocols and data algorithms [33]. VR and AR improve precision with intuitive feedback [12], [34], and force feedback enhances tactile perception [35]. However, challenges remain in maintaining system stability and minimizing delays under unstable network conditions [36], [37].

2.3 Simulation and Modeling in Robotics

Simulation plays a crucial role in robotic system development, allowing researchers to test design and control strategies in virtual environments, reducing costs, and minimizing hardware wear[38][39]. Tools like Gazebo provide realistic simulations of real-world effects, such as gravity and collisions, enabling validation of robot kinematics and dynamics[11][40]. RViz offers real-time visualization for robot states and sensor data, aiding in debugging and optimization, while CoppeliaSim supports multi-robot systems and complex scenarios with extensive customization options[10][41].

However, simulations often differ from real-world operations due to idealized assumptions, making it essential to bridge the gap between simulation and reality[42][43].

2.4 Conclusion and Gaps in Current Research

Although much progress has been made in robot teleoperation and simulation modeling, there are still some unsolved problems. For example, how to further reduce the network delay in teleoperation, improve the control accuracy, and enhance the operator's user experience remain important

directions for future research[44][45]. In addition, how to further reduce the difference between the simulation environment and the actual operation environment to improve the realistic adaptability of the simulation results is also a topic worthy of in-depth exploration[46][47].

This project will further explore the application of FR3 manipulator in telemanipulation based on existing research results. By modeling, simulating, and implementing remote control, we hope to bring new technological solutions to this field and provide strong support for future research.

3 Methods

This project aims to simulate Franka Emika's FR3 manipulator and implement remote control. The project is divided into key steps, each with detailed technical approaches and tools to ensure reliability and effectiveness.

3.1 URDF Wireless Transmission and Display

The first step of the project is to implement wireless transmission and display of URDF files for the FR3 manipulator. This will test the system's ability to share model data without data loss or corruption. The URDF file will be generated on the sending device, transferred wirelessly, and displayed on the receiving device using ROS and RViz[48].

The challenge is ensuring file integrity during transmission and seamless model display. Success will confirm the feasibility of remote data exchange and provide a foundation for future simulation and control operations, while also aiding in more complex data transfer tasks.

3.2 FR3 manipulator modelling and simulation

After successfully transferring and displaying the URDF file, we will proceed with the modeling and simulation of the FR3 manipulator using its xacro file. The xacro file simplifies the model's description, offering a more flexible and maintainable way to define the manipulator's structural and dynamic properties[49].

Two main simulation tools—RViz and Gazebo will be employed. RViz will display the manipulator's state and position in real time, providing an intuitive understanding of its behavior in the simulation environment. Gazebo, with its physics engine, will simulate the manipulator's physical movements and interactions in complex environments, making it ideal for verifying the manipulator's performance across different scenarios[50].

This stage aims to fully simulate the FR3 manipulator, ensuring that its model and behavior align closely with the actual device. This process is crucial for validating control strategies, identifying potential issues, and making early optimizations.

3.3 Real-time status monitoring and visualization

In addition to modeling and simulating the manipulator, this project will also facilitate real-time monitoring of its state. Utilizing RViz, we will display real-time data for each joint of the manipulator, including position, velocity, and acceleration. This information will be collected via the ROS system and displayed on a web interface, allowing users to monitor the manipulator's current operation anytime[51].

This enhancement will significantly increase system transparency and operability. The real-time status monitoring and visualization interface will enable operators to quickly assess the manipulator's condition and make necessary adjustments, both during setup and in live operation.

3.4 Remote Control Implementation

For remote control, this project will employ ROS master as the primary communication method. The device connected to the manipulator will send control commands via ROS, which will then be transmitted over the network as topics to another device. This receiving device will convert the topics into actionable control commands for the manipulator[52]. The ROS master-slave architecture ensures low-latency and high-reliability communication between devices.

The objective of this phase is to develop a stable, low-latency remote control system that ensures smooth operation of the FR3 manipulator across various network conditions. This will provide essential technical support for applications such as distributed manufacturing, remote maintenance, and medical surgery.

3.5 Motion Planning and Control

Motion planning and control is one of the core parts of this project. We will rely heavily on the MoveIt framework for the end-to-end control of the manipulator, which provides powerful motion planning algorithms, inverse kinematics solving, and collision detection, allowing us to realize path planning for the FR3 manipulator in complex environments[17].

In addition to MoveIt, we will also use Python code for more precise control. the flexibility of Python allows us to quickly write and test control strategies for a number of specific operational tasks[53]. Although code strategies and algorithm optimization are not the main object of research in this project, we will ensure that precise manipulation of the manipulator is achieved with these tools.

This part of the work will provide technical support for the application of manipulators in complex manipulation tasks and verify the effectiveness of different control methods.

4 Results and discussion

4.1 Implementation and effect of URDF wireless transmission

In the initial phase of this project, wireless transmission of URDF files via ROS master was implemented and the corresponding code was written via Python3 to handle the distribution and reception of the files. The system was designed to enable real-time transmission of URDF files between two devices via the ROS communication mechanism and visualization on the receiving device via RViz.

4.1.1 Overview of system implementation

The implementation of the system relies on the publishing and receiving mechanism of the ROS master node with the following workflow:

- Sending Device (Xacro Publisher): Converts the FR3 xacro file to URDF using ”xacro_publisher.py” and publishes the URDF data to the ROS master, which coordinates communication between ROS nodes.
- Receiving Device (Xacro Receiver): Runs ”xacro_receiver.py” to subscribe to the URDF data, then parses and displays the robot model in RViz. Both devices must configure ”ROS_MASTER_URI” and ”ROS_HOSTNAME” and verify communication with the ping command.

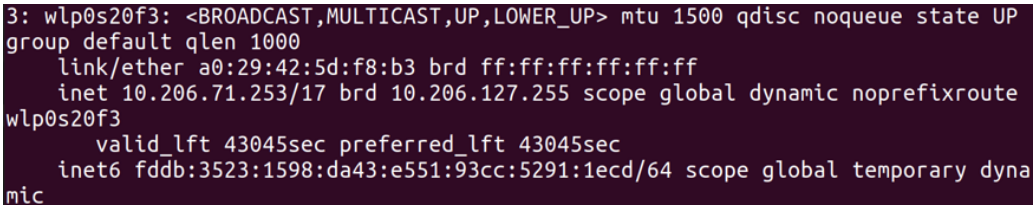

Fig. 6. Find IP of each device under the same network

4.1.2 System operation process

After setting up the ROS environment and installing Python dependencies, we started the ROS master on the sending device and ran ”xacro_publisher.py” to convert xacro files to URDF and create the publisher node. On the receiving device, ”xacro_receiver.py” connected to the ROS master and began receiving URDF data, which was verified by the tf tree (appendices D). The FR3 manipulator model was then loaded and displayed in RViz, enabling visualization of the manipulator’s joints and movements.

4.1.3 Presentation and discussion of results

During the experiment, the wireless transmission and display of URDF files was successfully achieved. The key results are shown and discussed below:

- Node Verification: Using ROS command line tools, we confirmed that the xacro_publisher node was active on the sending device and that the xacro_receiver node successfully subscribed to the specified ROS topic on the receiving device, with stable data transfer and no loss.
- RViz Display: The FR3 manipulator model displayed in RViz on the receiving device was structurally accurate, with smooth rotation and scaling, indicating efficient file transfer and parsing without significant delay.
- System Stability and Performance: Multiple experiments showed that under ideal network conditions, URDF files were transmitted with minimal delay, and the receiving device quickly displayed the model, confirming the system's real-time performance and reliability.

4.1.4 Summary

Through the experiments and implementations in this section, we successfully verified the feasibility of wireless URDF file transmission via ROS master. The Python scripts ensured the correct distribution and reception of the files, and the RViz display confirmed the system's reliability and practicality, laying the foundation for developing and testing remote control functions. For future improvements, we plan to optimize data compression and encryption during transmission to enhance adaptability in various network environments, and consider introducing error detection and recovery mechanisms to further improve system robustness.

4.2 Simulation and control of FR3 manipulators

In this section, we conducted simulation and control experiments on the FR3 manipulator. To achieve this, we first created the ”franka_simulation” package in the workspace and configured the relevant simulation files. In the simulation environment, we simulated the FR3 manipulator through ROS and Gazebo, and combined it with MoveIt's end controller to control the manipulator's operation. Next, we describe the implementation steps and results of the experiment in detail.

4.2.1 Creation and startup of the simulation environment

First, we created the ”franka_simulation” package in the workspace to manage the simulation files and configuration of the FR3 manipulator. To run the simulation, we launched the ”fr3_simulation.launch” file on one device. This launch file contains all the configurations needed to start the simulation and can open both the RViz and Gazebo simulation environments.

- Launch of RViz and Gazebo: Running the ”fr3_simulation.launch” file automatically starts RViz and Gazebo. The FR3 manipulator model is loaded into Gazebo, and accurately rendered in the virtual environment, while RViz provides a visual of the manipulator’s posture and joint status to aid in control.
- MoveIt Integration and Configuration: We manually created a MoveIt configuration for the FR3 manipulator, using MoveIt Assistant to define motion planning groups, end controllers, and kinematic solvers, enabling precise control of the manipulator’s end.

4.2.2 End effector control in RViz

After the simulation environment is started, we can control the end position of the manipulator through the MoveIt plug-in in RViz. MoveIt’s end controller allows users to adjust the position and attitude of the end of the manipulator intuitively in the RViz interface. The steps are as follows:

- Select Target Position: In RViz, users can adjust the manipulator’s end position and angle by dragging the target marker, enabling precise control of its movements in the simulation.
- Plan and Execute Motion: After setting the target, MoveIt generates an optimal, collision-free path for the manipulator. Users can then execute the motion, and the manipulator will follow the planned path to the target.

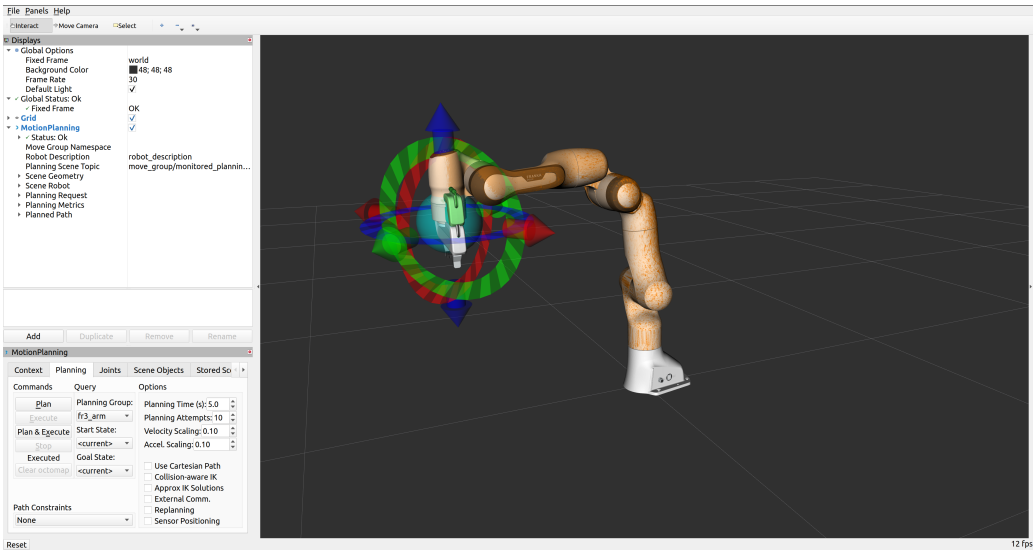

Fig. 7. End effector control simulation in RViz

4.2.3 Python Code Control

In addition to control via the MoveIt plugin in RViz, this project also supports direct control of the FR3 manipulator via Python scripts. By writing and running Python scripts, users can manipulate the movement of the manipulator more flexibly, especially when automated operations or complex control logic is required. Below is a description of the functionality of the Python script ”move_fr3.py” that we used in our experiments. The whole file content can be found in appendices E.

- The script begins by initializing the manipulator via MoveIt's API, starting the motion planning service, and connecting to the simulation model. It then retrieves the current end position using the 'get_current_pose()' function, which provides the end-effector's position and pose for motion planning.

```
moveit_commander.roscpp_initialize(sys.argv)
rospy.init_node('move_fr3', anonymous=True)
current_pose = move_group.get_current_pose().pose
rospy.loginfo("Current Pose: {}".format(current_pose))
```

- Set the target position: In our experiments, we moved the end of the manipulator forward by 10 cm (0.1 m). By setting a new target position based on the current end position, the script calls the set_pose_target() function to set the target position of the manipulator.

```
target_pose = geometry_msgs.msg.Pose()
target_pose.orientation = current_pose.orientation
target_pose.position.x = current_pose.position.x + 0.1
target_pose.position.y = current_pose.position.y
target_pose.position.z = current_pose.position.z
move_group.set_pose_target(target_pose)
```

- Plan and execution motion: After setting the target position, the script uses MoveIt's "go()" function to perform motion planning and execution. This function automatically calculates the optimal path and controls the manipulator to move according to the set target.

```
plan = move_group.plan()
if plan and plan[1]:
    rospy.loginfo("Executing plan...")
    move_group.execute(plan[1], wait=True)
```

- Print the new position and stop the manipulator: After the movement is complete, the script calls the get_current_pose() function again to get the new position of the manipulator and print it out to verify the accuracy of the movement.

```
move_group.stop()
move_group.clear_pose_targets()
new_pose = move_group.get_current_pose().pose
rospy.loginfo("New Pose: {}".format(new_pose))
rospy.sleep(2)
```

4.2.4 Experimental results and discussion

Through the above simulation and control operations, we have successfully implemented the simulation and control of the FR3 manipulator, and the following are the specific experimental results and discussions:

- Stability of the simulation environment: In the Gazebo simulation environment, the motion of the FR3 manipulator showed high stability and smoothness. The motion of the manipulator model in the simulation environment is consistent with the actual physical behavior, indicating the accuracy of the simulation model and controller configuration.
- Accuracy of end control: With the MoveIt plugin in RViz and Python scripts, we successfully controlled the end position of the manipulator. In the experiment, the manipulator was able to accurately follow the set target position without any path planning error or collision problem.
- Python Control Flexibility: The use of Python scripts greatly enhances control flexibility. With simple code modifications, users can quickly adjust the target position and motion path of the manipulator to automate operations. This approach is particularly suited to scenarios that require high-frequency control or specific task sequences.

4.2.5 Summary

The experiments demonstrate the feasibility of simulating and controlling the FR3 manipulator using ROS, Gazebo, RViz, and MoveIt. By customizing the MoveIt configuration, we achieved precise end-effector control, further enhanced by integrating Python code. The system exhibited good stability and responsiveness, providing a solid foundation for future real-world applications.

Future research can focus on optimizing the motion planning algorithm, reducing execution time, and exploring automation possibilities. Additionally, incorporating sensor feedback into control loops will be crucial for improving system adaptability in dynamic environments.

4.3 Remote control of the manipulator

In this section, we combine the simulation environment with a real FR3 manipulator to achieve remote control of the real manipulator. We connect to the manipulator through two devices with wired and wireless connections respectively, and use the network communication function of ROS to achieve remote manipulation of the manipulator between different devices. The specific implementation steps, results presentation, and discussion are given below.

4.3.1 System setup and device connection

In order to achieve remote control of the real FR3 manipulator, we first configured the hardware and software:

- Wired Connected Device (Master): This device is directly connected to the FR3 manipulator and controls it via ROS and the Franka Control Interface (FCI). After activating the manipulator, a launch file runs RViz (without Gazebo) to control the real manipulator. Users can manage the end position in RViz, similar to simulation, or use Python scripts with MoveIt's API for precise coordinate control.
- Wirelessly Connected Device (Slave): This device is connected to the master via Wi-Fi, this device receives key topics like "/tf" and "/joint_states" after setting up "ROS_MASTER_URI" and "ROS_HOSTNAME". It displays the real manipulator's state in RViz and allows end control through the RViz interface or Python scripts using MoveIt's API.

4.3.2 Main Configuration Files and Path Descriptions

In implementing the remote control process, we used the configuration package generated by MoveIt! Setup Assistant, which includes several key files that play an important role in implementing the control and visualization of the manipulator:

- Launch file
  - "fr3_moveit_config/launch/fr3.launch": this file is used to launch MoveIt! and RViz for visualization and testing on the slave device. This launch file allows us to load the model and controller of the manipulator in the RViz of the slave device and receive real-time attitude and state information from the master device.
  - "fr3_moveit_config/launch/move_group.launch": This file is used to launch the core functionality of MoveIt! including motion planning and control. This file is the basis for Python code control, and when launched enables MoveIt to communicate with the real manipulator and execute the planned motion.
- configuration file
  - "fr3_moveit_config/config/": This folder stores the kinematic configuration, planning parameters, and sensor configuration files for the manipulator. These configuration files define the manipulator's kinematic model, planning algorithms, joint constraints, and other information to ensure that the manipulator is able to perform operations in a real environment following the expected path.

4.3.3 Experimental process and results show

After successfully connecting the two devices, we performed the following tests and operations:

- Real-time Attitude Display and Control: Shown as Fig.8, in the RViz of the slave device, we can clearly see the attitude and joint status of the real manipulator. This data is transmitted

from the master device via Wi-Fi and updated in real-time, ensuring that the user has a comprehensive view of the status of the manipulator from the remote device. In the RViz interface of the slave device, we can adjust the end position of the manipulator by dragging the end controller, and the manipulator will make corresponding movements according to the control commands, with an effect similar to that of operating directly on the master device.

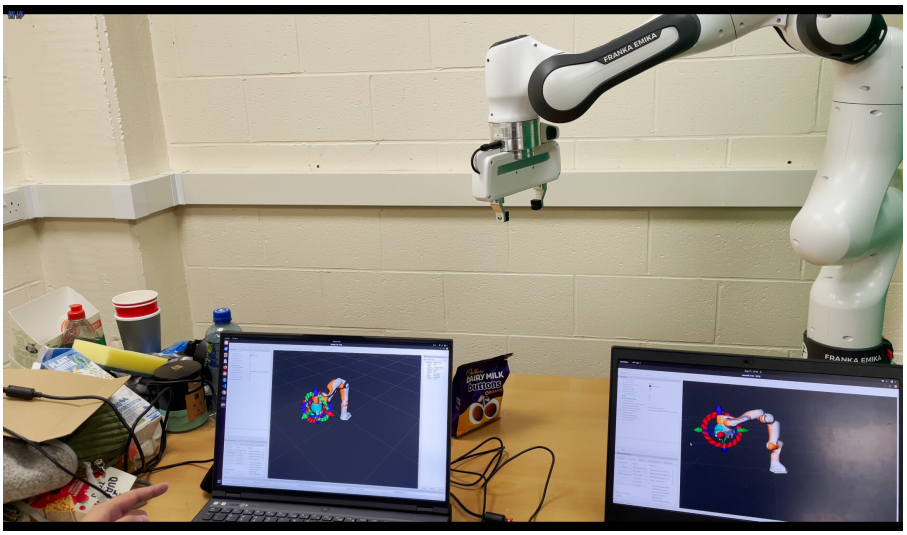

Fig. 8. Remote control through end effector

- Python code control: Shown as Fig.9, we ran a Python script on the slave device to control the end coordinates of the manipulator. The script implemented the adjustment of the end position through the MoveIt API and passed the control commands to the master device through the wireless network. The manipulator was able to move accurately according to the target position specified in the code, and the movement path was consistent with the planning, verifying the control accuracy and responsiveness of the system.

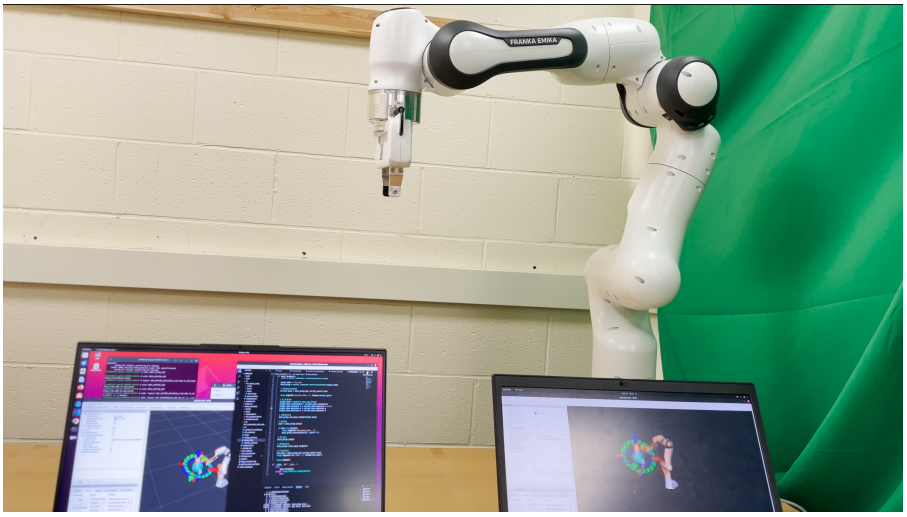

Fig. 9. Remote control through python code

4.3.4 System Performance and Discussion

Through the above experiments, we verified the performance of the remote control system of the real manipulator as follows:

- System Stability and Real-Time: The system demonstrated good stability in multiple tests. Under ideal network conditions, the control of the manipulator from the device has almost no delay, and the operation response is fast and accurate. At the same time, the real-time display in RViz was able to synchronously update the manipulator's attitude and joint status without any noticeable lag or delay.
- Impact of network environment: Under poor network conditions, we observed some delays in data transmission, resulting in a slight lag in the attitude update in RViz. Nevertheless, the execution of control commands remained stable and the movement of the manipulator was not significantly affected. This indicates that the system is somewhat resistant to network fluctuations, but there is still a need to optimize the efficiency of data transmission when the network is unstable.

4.3.5 Summary

By experimenting with remote control of a real FR3 manipulator from two devices, we have successfully verified the feasibility of the system in practice. Both on the local and remote devices, users can precisely control the manipulator through RViz and Python codes. The system demonstrated good real-time performance, stability, and control accuracy, making it suitable for industrial and scientific application scenarios that require remote operation.

Future work may include further optimization of network conditions, especially in unstable or high-latency network environments, to enhance the robustness of the system. Meanwhile, closed-loop control incorporating sensor feedback will help improve the system's adaptability and operational accuracy in dynamic environments.

5 Conclusions and outlook

5.1 Conclusion

This project successfully realized the control and remote operation of the FR3 manipulator in both simulated and real environments and achieved significant results in three key areas.

1. The wireless transfer of URDF files effectively facilitates sharing manipulator models across remote devices. Utilizing ROS communication, we successfully transferred URDF files between devices and demonstrated the manipulator model in RViz, validating the feasibility of cross-device model sharing and setting the stage for remote control.

2. For the FR3 manipulator simulation and control, we configured and ran the simulation environment, using the ”fr3_simulation.launch” file to load the model and integrate it with MoveIt for precise end-effector control. Python scripting further enhanced manipulation flexibility, indicating the system’s effectiveness in automating complex tasks and its operational feasibility in a virtual setting.

3. In the wireless control experiment of the real FR3 manipulator, we achieved remote control through a cooperative setup between two devices—one connected to the manipulator as the ROS master and the other linked via Wi-Fi for topic reception and processing. This setup enabled remote monitoring and control via RViz and Python, demonstrating the system’s capability in real-world operation and affirming the reliability of ROS and MoveIt for remote tasks.

5.2 Outlook

Despite the significant results of the project, there are still many directions that can be improved and optimized for the wireless control of real manipulators. Future work will focus on the following areas:

1. System stability optimization: The current system performs well under ideal network conditions, but real-time control and data transmission reliability may degrade in unstable or high-latency networks. Future work could focus on implementing efficient communication protocols or data compression to reduce latency and packet loss, along with robust error handling and data recovery mechanisms to enhance stability in complex network environments.

2. Enhanced monitoring platform: While the system currently uses RViz for status display and control, future improvements could involve developing a more robust and user-friendly monitoring platform. This could include incorporating additional sensor data, such as vision and force feedback, and possibly using a real-time web-based platform or VR technology to provide more comprehensive and intuitive monitoring and manipulation capabilities.

3. Advanced data presentation and analysis: Presently, data is primarily presented through RViz and command line output. Future work could focus on developing advanced tools for real-time graphical display of key performance indicators like speed, accuracy, and latency. This would assist operators in optimizing the system and serve as a critical resource for troubleshooting and performance tuning.

4. Expanded application scenario testing: Beyond the basic control functions, future testing could explore the system’s performance in more complex tasks, such as collaborative robot operations in industrial production or telesurgery. Testing in these challenging scenarios would further validate the system’s stability and practical usefulness.

With the above optimizations and extensions, the project will be able to provide a more efficient and reliable solution in the remote operation of real manipulators, opening up wider prospects for industrial and scientific applications.

References

[1] B. M. Yilmaz, E. Tatlicioglu, A. Savran, and M. Alci, “Adaptive fuzzy logic with self-tuned membership functions based repetitive learning control of robotic manipulators,” *Applied Soft Computing*, vol. 104, p. 107 183, 2021 (cited on p. 9).

[2] S. Chatterjee, S. Das, K. Ganguly, and D. Mandal, “Advancements in robotic surgery: Innovations, challenges and future prospects,” *Journal of Robotic Surgery*, vol. 18, no. 1, p. 28, 2024 (cited on p. 9).

[3] F. Emika, *Official website of franka research 3*, [Online], `https://franka.de/research`, 2024 (cited on p. 9).

[4] gazebo, *Qviro, research 3 robot reviews*, [Online], `https://qviro.com/product/franka-emika/research3`, 2024 (cited on p. 9).

[5] J. Pliego-Jiménez, M. A. Arteaga-Pérez, and C. Cruz-Hernández, “Dexterous remote manipulation by means of a teleoperation system,” *Robotica*, vol. 37, no. 8, pp. 1457–1476, 2019 (cited on p. 9).

[6] L. Žlajpah, “Simulation in robotics,” *Mathematics and Computers in Simulation*, vol. 79, no. 4, pp. 879–897, 2008 (cited on p. 9).

[7] C. S. Tzafestas, “Web-based laboratory on robotics: Remote vs. virtual training in programming manipulators,” *Web-Based Control and Robotics Education*, pp. 195–225, 2009 (cited on p. 10).

[8] Stanford Artificial Intelligence Laboratory et al., *Robotic operating system*, version ROS Melodic Morenia, May 23, 2018. [Online]. Available: `https://www.ros.org` (cited on p. 10).

[9] A. Robotics, *Getting ready to build a ros robot*, [Online], `https://articulatedrobotics.xyz/tutorials/ready-for-ros/urdf/`, 2024 (cited on p. 10).

[10] RViz, *Official website of rviz*, [Online], `https://wiki.ros.org/rviz`, 2024 (cited on pp. 11, 12).

[11] gazebo, *Official website of gazebo*, [Online], `https://gazebosim.org/home`, 2024 (cited on pp. 11, 12).

[12] A. Assila, A. Dhouib, Z. Monla, and M. Zghal, “Integration of augmented, virtual and mixed reality with building information modeling: A systematic review,” in *International Conference on Human-Computer Interaction*, Springer, 2022, pp. 3–19 (cited on pp. 11, 12).

[13] F. Sanfilippo, Ø. Stavdahl, and P. Liljebäck, “Snakesim: A ros-based rapid-prototyping framework for perception-driven obstacle-aided locomotion of snake robots,” in *SnakeSIM: a ROS-based control and simulation framework for perception-driven obstacle-aided locomotion of snake robots*, Dec. 2017. DOI: 10.1109/ROBIO.2017.8324585 (cited on p. 11).

[14] D. Coleman, I. Sucan, S. Chitta, and N. Correll, “Reducing the barrier to entry of complex robotic software: A moveit! case study,” *arXiv preprint arXiv:1404.3785*, 2014 (cited on p. 11).

[15] Anonymous, “Reducing the barrier to entry of complex robotic software: A moveit! case study,” *arXiv*, 2024. [Online]. Available: https://ar5iv.org/abs/1404.3785 (cited on p. 11).

[16] S. Chitta, *MoveIt!: An Introduction* (Studies in Computational Intelligence). Springer, Cham, 2016, vol. 625, pp. 3–27. DOI: 10.1007/978-3-319-26054-9_1. [Online]. Available: https://doi.org/10.1007/978-3-319-26054-9_1 (cited on p. 11).

[17] R. Wehner, “Introducing moveit pro: Rapid robotics application development for unstructured environments,” *MoveIt*, 2024. [Online]. Available: https://moveit.ros.org (cited on pp. 11, 14).

[18] K. Selvaraj, S. Maruthachalam, and B. Veerasamy, “Kinematic modeling and performance analysis of a 5-dof robot for welding applications,” *Machines*, vol. 12, no. 6, p. 378, 2024. DOI: 10.3390/machines12060378. [Online]. Available: https://www.mdpi.com/2075-1702/12/6/378 (cited on p. 11).

[19] J. Kofman, X. Wu, T. J. Luu, and S. Verma, “Teleoperation of a robot manipulator using a vision-based human-robot interface,” *IEEE Transactions on Industrial Electronics*, vol. 52, no. 5, pp. 1206–1219, 2005. DOI: 10.1109/TIE.2005.855673 (cited on p. 11).

[20] P. Kah, M. Shrestha, E. Hiltunen, and J. Martikainen, “Robotic arc welding sensors and programming in industrial applications,” *International Journal of Mechanical and Materials Engineering*, vol. 10, pp. 1–16, 2015 (cited on p. 11).

[21] M. Blatnickỳ, J. Dižo, J. Gerlici, M. Sága, T. Lack, and E. Kuba, “Design of a robotic manipulator for handling products of automotive industry,” *International Journal of Advanced Robotic Systems*, vol. 17, no. 1, p. 1 729 881 420 906 290, 2020 (cited on p. 11).

[22] G. C. Zutin, E. C. Pulquerio, A. V. Pasotti, G. F. Barbosa, and S. B. Shiki, “Application of robotic manipulator technology and its relation to additive manufacturing process—a review,” *The International Journal of Advanced Manufacturing Technology*, pp. 1–15, 2024 (cited on p. 11).

[23] J. Kofman, X. Wu, T. J. Luu, and S. Verma, “Teleoperation of a robot manipulator using a vision-based human-robot interface,” *IEEE transactions on industrial electronics*, vol. 52, no. 5, pp. 1206–1219, 2005 (cited on p. 12).

[24] Z. Wang, M. Hai, X. Liu, Z. Pei, S. Qian, and D. Wang, “A human–robot interaction control strategy for teleoperation robot system under multi-scenario applications,” *International Journal of Intelligent Robotics and Applications*, pp. 1–21, 2024 (cited on p. 12).

[25] A. Munawar and G. Fischer, “A surgical robot teleoperation framework for providing haptic feedback incorporating virtual environment-based guidance,” *Frontiers in Robotics and AI*, vol. 3, p. 47, 2016. DOI: 10.3389/frobt.2016.00047 (cited on p. 12).

[26] N. Feizi, M. Tavakoli, R. V. Patel, and S. F. Atashzar, “Robotics and ai for teleoperation, tele-assessment, and tele-training for surgery in the era of covid-19: Existing challenges, and future vision,” *Frontiers in Robotics and AI*, vol. 8, p. 610 677, 2021 (cited on p. 12).

[27] T. Q. Dinh, J. I. Yoon, C. Ha, J. Marco, and K. Jian, “A real-time bilateral teleoperation control system over imperfect network,” in *Real-time Systems*, InTech Rijeka, 2016, pp. 117–140 (cited on p. 12).

[28] S. Robotic, *Arm - remote control*, [Online], https://docs.sunfounder.com/projects/piarm/zh/latest/ezblock/remote_control_ez.html#ezb-arm-remote, 2024 (cited on p. 12).

[29] A. N. Cundar, R. Fotouhi, Z. Ochitwa, and H. Obaid, “Quantifying the effects of network latency for a teleoperated robot,” *Sensors*, vol. 23, no. 20, p. 8438, 2023. DOI: 10.3390/s23208438. [Online]. Available: https://www.mdpi.com/1424-8220/23/20/8438 (cited on p. 12).

[30] Y. Su, L. Lloyd, X. Chen, and J. G. Chase, “Latency mitigation using applied hmms for mixed reality-enhanced intuitive teleoperation in intelligent robotic welding,” *The International Journal of Advanced Manufacturing Technology*, vol. 126, no. 5, pp. 2233–2248, 2023 (cited on p. 12).

[31] X. Na, Y. Zhan, Y. Xia, and J. Dong, “Control of networked systems with packet loss and channel uncertainty,” *IET Control Theory & Applications*, vol. 10, no. 17, pp. 2251–2259, 2016 (cited on p. 12).

[32] M.-Y. Zhao, H.-P. Liu, Z.-J. Li, and D.-H. Sun, “Fault tolerant control for networked control systems with packet loss and time delay,” *International Journal of Automation and Computing*, vol. 8, no. 2, pp. 244–253, 2011 (cited on p. 12).

[33] E. Mozaffariahrar, F. Theoleyre, and M. Menth, “A survey of wi-fi 6: Technologies, advances, and challenges,” *Future Internet*, vol. 14, no. 10, p. 293, 2022. DOI: 10.3390/fi14100293. [Online]. Available: https://www.mdpi.com/1999-5903/14/10/293 (cited on p. 12).

[34] M. Gallipoli, S. Buonocore, M. Selvaggio, G. A. Fontanelli, S. Grazioso, and G. Di Gironimo, “A virtual reality-based dual-mode robot teleoperation architecture,” *Robotica*, pp. 1–24, 2024 (cited on p. 12).

[35] J. H. Bong, S. Choi, J. Hong, and S. Park, “Force feedback haptic interface for bilateral teleoperation of robot manipulation,” *Microsystem Technologies*, vol. 28, no. 10, pp. 2381–2392, 2022 (cited on p. 12).

[36] Y. Wang, J. Tian, Y. Liu, *et al.*, “Adaptive neural network control of time delay teleoperation system based on model approximation,” *Sensors*, vol. 21, no. 22, p. 7443, 2021 (cited on p. 12).

[37] S. Su and Y. Ji, “Fixed-time adaptive neural network synchronization control for teleoperation system with position error constraints and time-varying delay,” *Nonlinear Dynamics*, vol. 111, no. 14, pp. 13 053–13 072, 2023 (cited on p. 12).

[38] Anonymous, “What is robot simulation?” *MathWorks*, 2022. [Online]. Available: `https://www.mathworks.com/discovery/robot-simulation.html` (cited on p. 12).

[39] U. Rastogi, J. Sayyad, B. Ramesh, and A. Bongale, “Improved accuracy of robotic arm using virtual environment,” in *International Conference on Artificial Intelligence on Textile and Apparel*, Springer, 2023, pp. 95–108 (cited on p. 12).

[40] S. Hammad and S. A. Maged, “Kinematic and dynamic modeling of 3dof variable stiffness links manipulator with experimental validation,” *Applied Sciences*, vol. 14, no. 12, p. 5285, 2024. DOI: `10.3390/app14125285`. [Online]. Available: `https://www.mdpi.com/2075-1702/12/6/378` (cited on p. 12).

[41] Anonymous, “Roboscript: Code generation for free-form manipulation tasks across real and simulation,” *Arxiv*, 2023. [Online]. Available: `https://arxiv.org/abs/2402.14623` (cited on p. 12).

[42] Y. Xu, H. Mellmann, and H.-D. Burkhard, “An approach to close the gap between simulation and real robots,” in *Simulation, Modeling, and Programming for Autonomous Robots: Second International Conference, SIMPAR 2010, Darmstadt, Germany, November 15-18, 2010. Proceedings 2*, Springer, 2010, pp. 533–544 (cited on p. 12).

[43] K. Bousmalis and S. Levine, “Closing the simulation-to-reality gap for deep robotic learning,” *Google Research Blog*, vol. 1, 2017 (cited on p. 12).

[44] Y. Zhu, K. Fusano, T. Aoyama, and Y. Hasegawa, “Intention-reflected predictive display for operability improvement of time-delayed teleoperation system,” *ROBOMECH Journal*, vol. 10, no. 1, p. 17, 2023 (cited on p. 13).

[45] P. Farajiparvar, H. Ying, and A. Pandya, “A brief survey of telerobotic time delay mitigation,” *Frontiers in Robotics and AI*, vol. 7, p. 578 805, 2020 (cited on p. 13).

[46] K. M. Adams, *Adaptability, Flexibility, Modifiability and Scalability, and Robustness*. Springer, 2015. DOI: 10.1007/978-3-319-18344-2_9. [Online]. Available: https://link.springer.com/book/10.1007/978-3-319-18344-2 (cited on p. 13).

[47] K. Weersink, A. K. Hall, J. Rich, A. Szulewski, and J. D. Dagnone, "Simulation versus real-world performance: A direct comparison of emergency medicine resident resuscitation entrustment scoring," *Advances in Simulation*, vol. 4, pp. 1–10, 2019 (cited on p. 13).

[48] W. K. Chung, L.-C. Fu, and T. Kröger, "Motion control," *Springer handbook of robotics*, pp. 163–194, 2016 (cited on p. 13).

[49] S. A. I. L. et al., *Using xacro to clean up your code*, [Online], https://docs.ros.org/en/humble/Tutorials/Intermediate/URDF/Using-Xacro-to-Clean-Up-a-URDF-File.html, 2024 (cited on p. 13).

[50] T. Wright, A. West, M. Licata, N. Hawes, and B. Lennox, "Simulating ionising radiation in gazebo for robotic nuclear inspection challenges," *Robotics*, vol. 10, no. 3, p. 86, 2021 (cited on p. 13).

[51] P. Siagian and K. Shinoda, "Web based monitoring and control of robotic arm using raspberry pi," in *2015 International Conference on Science in Information Technology (ICSITech)*, IEEE, 2015, pp. 192–196 (cited on p. 14).

[52] Z. Ren, C. Chen, and J. Fu, "Mobile edge computing: Low latency and high reliability," in *Encyclopedia of Wireless Networks*, Springer, 2020, pp. 873–877 (cited on p. 14).

[53] M. Quigley, B. Gerkey, and W. D. Smart, *Programming Robots with ROS: a practical introduction to the Robot Operating System*. " O'Reilly Media, Inc.", 2015 (cited on p. 14).

Appendices

A Project outline

A.1 Project Scope

A.1.1 Project Overview

- Introduction to Robotics and Remote Control: Discuss the growing importance of robotics and automation in modern technology and various industries.
- Background on Franka Research 3: Provide an overview of the Franka Research 3 robotic arm, emphasizing its advanced features and significance in current research.
- Current State of Remote Control Technologies: Examine the need for user-friendly interfaces in complex robotic systems, establishing the relevance of the project.

A.1.2 Function of Device

- Franka Research 3 Specifications: Detail the design, key features, and potential applications of the robotic arm.
- Web Interface for Remote Control: Describe the user interface/experience considerations, the technologies used, and how the interface facilitates intuitive control.
- Integration for Real-Time Control: Explain the process of ensuring accurate, reliable, and real-time control through the web interface.

A.1.3 Project Stages

- Planning Phase: Outline initial research, objective setting, and requirement definition.
- Development Phase: Describe the design, coding, and integration of the web interface with the robotic arm.
- Testing Phase: Summarize the testing methodologies and results.
- Deployment Phase: Discuss the system deployment, user feedback, and final adjustments.

A.1.4 Project Constraints

- Technical Constraints: Address hardware limitations and software compatibility issues.
- Time Constraints: Provide a timeline, key milestones, and how time limitations influenced the project.

A.2 Project Motivation

- Academic Contributions: Explain how the project contributes to the field of robotics and aligns with academic goals.
- Industry Demand and Practical Implications: Explore the demand for advanced robotic control systems in various industries and the potential impact of the project.

A.3 Project Purpose

- Research Purpose: Focus on developing a functional remote control system for robotic arms, testing the feasibility and effectiveness of a web-based interface.
- Practical Purpose: Emphasize creating a user-friendly interface for the Franka Research 3 to enhance accessibility, usability, and broader application potential.

A.4 Project Objectives

A.4.1 Primary Objectives

- Ensure Real-Time Control: Achieve responsive and precise control.
- Maintain Accuracy: Ensure tasks are performed with high precision.
- Enabling remote control: remote control of manipulators by means of wireless communication.

A.4.2 Secondary Objectives

- Document Process: Provide comprehensive documentation of the development.
- Share Findings: Publish results to contribute to academia.
- Identify Improvements: Continuously evaluate and refine the system.
- Explore Future Developments: Investigate additional features and potential applications.

B Risk assessment

MANCHESTER 1824

The University of Manchester

General Risk Assessment Form

Date: (1) 06/29/2024	Assessed by: (2) Wenjie Zhao	Checked / Validated* by: (3) Wei Pan	Location: Home and University of Manchester(4)	Assessment ref no (5)	Review date: (6)
Task/premises: (7) Working from home and from university for final dissertation --- Digital twins Franka research 3					

Activity (8)	Hazard (9)	Who might be harmed and how (10)	Existing measures to control risk (11)	Risk rating (12)	Result (13)
Working from home	Lone working	Wenjie Zhao Isolated	1. Regular check-ins, either through phone calls or electronic means, to ensure the lone worker's safety.	LOW	A
Working from home And Working at office	long periods looking at DSE (display screen equipment)	Wenjie Zhao Eye strain. Back strain	1. Encourage regular breaks away from the screen, such as every 40 minutes take a 5-minute break. 2. Adjust screen settings, such as brightness, contrast, and font size, to minimize glare and make it easier to read without straining the eyes. 3. Using proper lighting to reduce eye strain.	LOW	A
Working from home	Stress / Wellbeing	Wenjie Zhao Psychosocial	1. Flexible working arrangements, such as flexible hours, balance dissertation and personal life.	LOW	A

Result : T = trivial, A = adequately controlled, N = not adequately controlled, action required, U = unknown risk

Activity (8)	Hazard (9)	Who might be harmed and how (10)	Existing measures to control risk (11)	Risk rating (12)	Result (13)
Use of electrical appliances	Misuse of electrical appliance	Wenjie Zhao Electric shock, Fire	1. All equipment used in accordance with the manufacture 2. Visual checks before use to make sure equipment free from defects	MED	A
Working from home	Illness	Wenjie Zhao Flu	1. Find a personal GP and report to the tutor 2. Report to the School Safety Advisor to complete an accident/incident form	Low	A
Working at the University of Manchester	Fire Safety	Wenjie Zhao Injury or fatality from fire, smoke inhalation.	1. Regularly check surroundings to ensure that fire exits are not blocked and report any obstructions immediately. 2. Report any hazards or concerns related to fire safety to my supervisor or the safety officer promptly.	Low	A

Result : T = trivial, A = adequately controlled, N = not adequately controlled, action required, U = unknown risk

Action plan (14)				
Ref No	**Further action required**	**Action by whom**	**Action by when**	**Done**

Result : T = trivial, A = adequately controlled, N = not adequately controlled, action required, U = unknown risk

Notes to accompany General Risk Assessment Form

This form is the one recommended by Health & Safety Services, and used on the University's risk assessment training courses. It is strongly suggested that you use it for all new assessments, and when existing assessments are being substantially revised. However, its use is not compulsory. Providing the assessor addresses the same issues; alternative layouts may be used.

(1) **Date** : Insert date that assessment form is completed. The assessment must be valid on that day, and subsequent days, unless circumstances change and amendments are necessary.

(2) **Assessed by** : Insert the name and signature of the assessor. For assessments other than very simple ones, the assessor should have attended the University course on risk assessments (link to STDU)

(3) **Checked / Validated* by** : delete one.

Checked by : Insert the name and signature of someone in a position to check that the assessment has been carried out by a competent person who can identify hazards and assess risk, and that the control measures are reasonable and in place. The checker will normally be a line manager, supervisor, principal investigator, etc. Checking will be appropriate for most risk assessments.

Validated by : Use this for higher risk scenarios, eg where complex calculations have to be validated by another "independent" person who is competent to do so, or where the control measure is a strict permit-to-work procedure requiring thorough preparation of a workplace. The validator should also have attended the University's risk assessment course or equivalent, and will probably be a chartered engineer or professional with expertise in the task being considered. Examples of where validation is required include designs for pressure vessels, load-bearing equipment, lifting equipment carrying personnel or items over populated areas, and similar situations.

(4) **Location** : insert details of the exact location, ie building, floor, room or laboratory etc

(5) **Assessment ref no** : use this to insert any local tracking references used by the school or administrative directorate

(6) **Review date** : insert details of when the assessment will be reviewed as a matter of routine. This might be in 1 year's time, at the end of a short programme of work, or longer period if risks are known to be stable. Note that any assessment must be reviewed if there are any significant changes – to the work activity, the vicinity, the people exposed to the risk, etc

(7) **Task / premises** : insert a brief summary of the task, eg typical office activities such as filing, DSE work, lifting and moving small objects, use of misc electrical equipment. Or, research project [title] involving the use of typical laboratory hardware, including fume cupboards, hot plates, ovens, analysis equipment, flammable solvents, etc.

(8) **Activity** : use the column to describe each separate activity covered by the assessment. The number of rows is unlimited, although how many are used for one assessment will depend on how the task / premises is sub-divided. For laboratory work, activities in one particular lab or for one particular project might include; use of gas cylinders, use of fume cupboard, use of computer or other electrical equipment, use of lab ovens, hot plates or heaters, use of substances hazardous to health, etc

(9) **Hazard** : for each activity, list the hazards. Remember to look at hazards that are not immediately obvious. For example, use of a lathe will require identification of the machine hazards, but also identification of hazards associated with the use of cutting oils (dermatitis), poor lighting, slipping on oil leaks, etc. The same activity might well have several hazards associated with it. Assessment of simple chemical risks (eg use of cleaning chemicals in accordance with the instructions on the bottle) may be recorded here. More complex COSHH

assessments eg for laboratory processes, should be recorded on the specific COSHH forms (link).

(10) **Who might be harmed and how** : insert everyone who might be affected by the activity and specify groups particularly at risk. Remember those who are not immediately involved in the work, including cleaners, young persons on work experience, maintenance contractors, Estates personnel carrying out routine maintenance and other work. Remember also that the risks for different groups will vary. Eg someone who needs to repair a laser may need to expose the beam path more than users of the laser would do. Vulnerable groups could include children on organised visits, someone who is pregnant, or employees and students with known disabilities or health conditions (this is not a definitive list).

For each group, describe how harm might come about, eg an obstruction or wet patch on an exit route is a hazard that might cause a trip and fall; use of electrical equipment might give rise to a risk of electric shock; use of a ultraviolet light source could burn eyes or skin.

(11) **Existing measures to control the risk** : list all measures that already mitigate the risk. Many of these will have been implemented for other reasons, but should nevertheless be recognised as means of controlling risk. For example, restricting access to laboratories or machine rooms for security reasons also controls the risk of unauthorised and unskilled access to dangerous equipment. A standard operating procedure or local rules (eg for work with ionising radiation, lasers or biological hazards) will often address risks. Some specific hazards may require detailed assessments in accordance with specific legislation (eg COSHH, DSEAR, manual handling, DSE work). Where this is the case, and a detailed assessment has already been done in another format, the master risk assessment can simply cross-reference to other documentation. For example, the activity might be use of a carcinogen, the hazard might be exposure to hazardous substances, the existing control measures might all be listed in a COSHH assessment. Controls might also include use of qualified and/or experienced staff who are competent to carry out certain tasks; an action plan might include training requirements for other people who will be carrying out those tasks.

(12) **Risk Rating** : the simplest form of risk assessment is to rate the remaining risk as high, medium or low, depending on how likely the activity is to cause harm and how serious that harm might be.

The risk is **LOW** - if it is most unlikely that harm would arise under the controlled conditions listed, and even if exposure occurred, the injury would be relatively slight.
The risk is **MEDIUM** - if it is more likely that harm might actually occur and the outcome could be more serious (eg some time off work, or a minor physical injury.
The risk is **HIGH** - if injury is likely to arise (eg there have been previous incidents, the situation looks like an accident waiting to happen) and that injury might be serious (broken bones, trip to the hospital, loss of consciousness), or even a fatality.

Schools or administrative directorates may choose to use other rating systems. Typical amongst these are matrices (of 3x3, 4x4, 5x5 or even more complex) which require the assessor to select a numerical rating for both "likelihood that harm will arise" and "severity of that harm". These may give a spurious sense of accuracy and reliability – none are based on quantitative methods. There are methods of estimating risk quantitatively, and these may be appropriate for complex design of load bearing structures and the like. Advice on methods of risk assessment is available from HSS. Whatever system of assessment is adopted, it is **essential** that the assessor has received suitable training and is familiar with the meaning of the terms (or numbers) used.

(13) **Result** : this stage of assessment is often overlooked, but is probably the most important. Assigning a number or rating to a risk does not mean that the risk is necessarily adequately controlled. The options for this column are:

T = trivial risk. Use for very low risk activities to show that you have correctly identified a hazard, but that in the particular circumstances, the risk is insignificant.

A = adequately controlled, no further action necessary. If your control measures lead you to conclude that the risk is low, and that all legislative requirements have been met (and University policies complied with), then insert A in this column.

N = not adequately controlled, actions required. Sometimes, particularly when setting up new procedures or adapting existing processes, the risk assessment might identify that the risk is high or medium when it is capable of being reduced by methods that are reasonably practicable. In these cases, an action plan is required. The plan should list the actions necessary, who they are to be carried out by, a date for completing the actions, and a signature box for the assessor to sign off that the action(s) has been satisfactorily completed. Some action plans will be complex documents; others may be one or two actions that can be completed with a short timescale.

U = unable to decide. Further information required. Use this designation if the assessor is unable to complete any of the boxes, for any reason. Sometimes, additional information can be obtained readily (eg from equipment or chemicals suppliers, specialist University advisors) but sometimes detailed and prolonged enquiries might be required. Eg is someone is moving a research programme from a research establishment overseas where health and safety legislation is very different from that in the UK.

For T and A results, the assessment is complete.
For N or U results, more work is required before the assessment can be signed off.

(14) **Action Plan**. Include details of any actions necessary in order to meet the requirements of the information in Section 11 'Existing measures to control the risk'. Identify someone who will be responsible for ensuring the action is taken and the date by which this should be completed. Put the date when the action has been completed in the final column.

C Work on github

All the work content including original files and related images and video can be found at fr3_control.

D TF tree

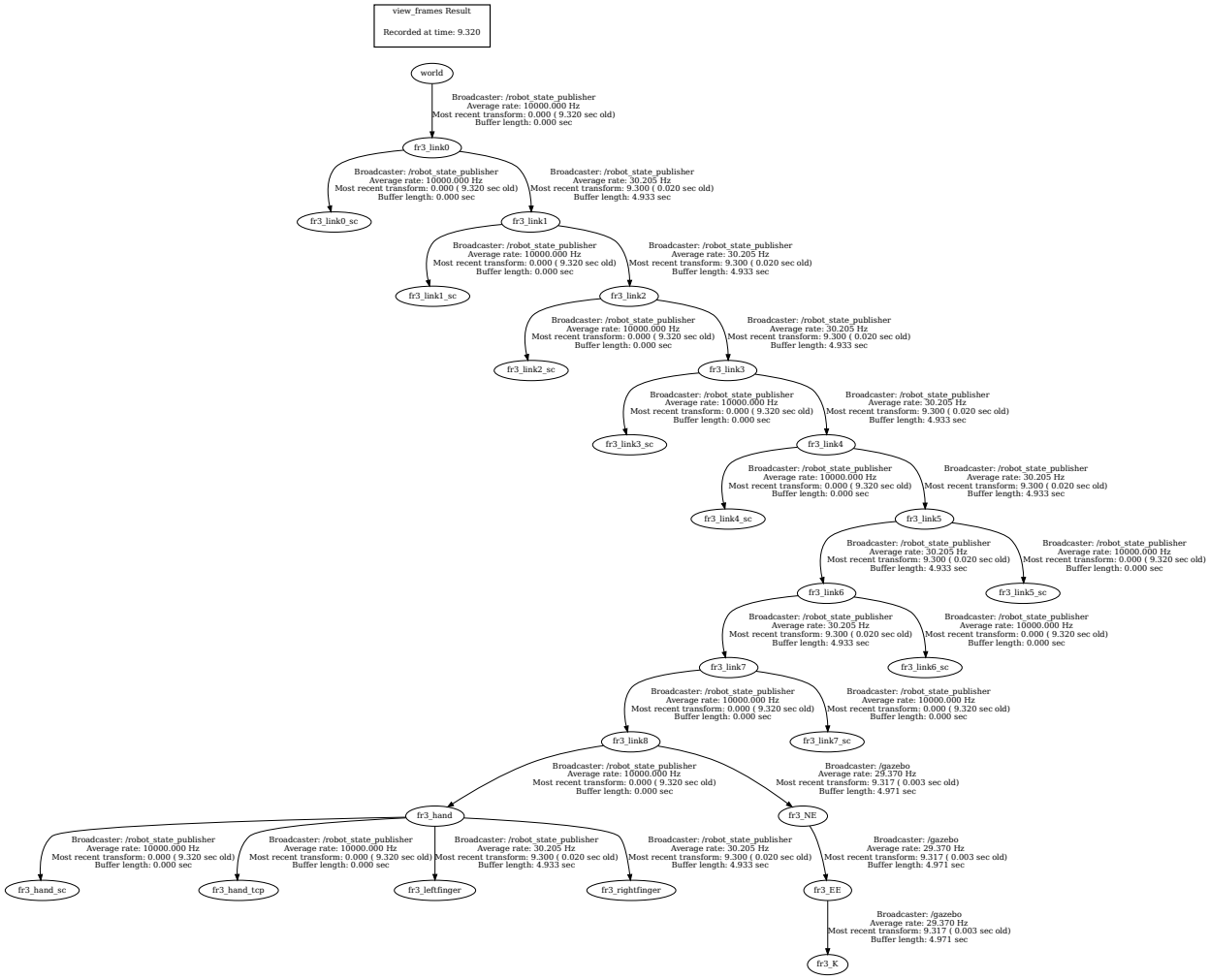

Fig. 10. The TF tree of the manipulator

E Python control code

```
#!/usr/bin/env python

import sys
import rospy
import moveit_commander
import geometry_msgs.msg
```

```
def move_forward():
    # Initialize the MoveIt API
    moveit_commander.roscpp_initialize(sys.argv)
    rospy.init_node('move_fr3', anonymous=True)

    # Robot interface
    robot = moveit_commander.RobotCommander()
    scene = moveit_commander.PlanningSceneInterface()

    group_name = "fr3_arm"
    move_group = moveit_commander.MoveGroupCommander(group_name)

    # Get the current end position
    current_pose = move_group.get_current_pose().pose

    rospy.loginfo("Current Pose: {}".format(current_pose))

    # Define the target position
    target_pose = geometry_msgs.msg.Pose()
    target_pose.orientation = current_pose.orientation
    target_pose.position.x = current_pose.position.x - 0.1
    target_pose.position.y = current_pose.position.y
    target_pose.position.z = current_pose.position.z

    # Set the target pose
    move_group.set_pose_target(target_pose)

    # Plan the path
    plan = move_group.plan()

    if plan and plan[1]:
        rospy.loginfo("Executing plan...")
        move_group.execute(plan[1], wait=True)

    # Stop the movement
    move_group.stop()

    # Clear the target pose
    move_group.clear_pose_targets()

    # Print the new pose information
    new_pose = move_group.get_current_pose().pose
    rospy.loginfo("New Pose: {}".format(new_pose))
```

```
    rospy.sleep(2)

if __name__ == "__main__":
    try:
        move_forward()
    except rospy.ROSInterruptException:
        pass
```