Real-Time VR Teleoperation of a Franka Arm for Construction Automation and Human-Robot Interaction

Arpit Jain

Supervisors: Yuezhen Gao, Dr.Qipei Mei

Civil and Environmental Engineering Department, University of Alberta

Overview

This document serves as a comprehensive technical guide and handover manual for the real-time XR telecontrol system built during the internship project. It includes system architecture, detailed setup steps, terminal commands, software configurations, code references, and annotated screenshots.

Contents

1 Introduction **3**

2 System Architecture **3**

3 Key Technologies **3**

4 Hardware Setup and Networking **3**

5 Franka Robot Setup **4**

6 ROS 2 Packages **4**

7 Unity Application **5**

8 Networking **5**

9 Execution Workflow **5**

10 Running Custom Code (`libfranka`) **5**

11 ALVR + SteamVR Setup on Linux for Meta Quest 2 (USB-C Connection) **6**

12 Run Instructions: Full System Startup After Setup **7**

13 Franak UI Instructions **11**

14 Troubleshooting **26**

15 Results and Impact **26**

16 Future Work **27**

17 Additional Links **27**

18 Acknowledgements **27**

1 Introduction

- Tele-operated robotic systems often rely on Windows-only VR toolchains, introducing high latency and limited cross-platform compatibility.
- This project demonstrates a fully Linux-compatible pipeline using Meta Quest 2 to control a Franka Research 3 robotic arm in real time (less than 20 ms end-to-end latency).
- The system is open-source, reproducible, and modular.

2 System Architecture

- Meta Quest 2 (controller tracking only)
- ALVR Client + ALVR Server
- SteamVR Runtime
- Unity App with ROS-TCP Connector
- ROS-TCP-Endpoint (Jetson Nano)
- ROS 2 Nodes:
  - `franka_cartesian_control`
  - `franka_vr_control`
- ROS 2 Humble (Jetson RT Kernel)
- Franka Research 3 Robot Arm

3 Key Technologies

- Meta Quest 2, ALVR & SteamVR, Unity 2022
- ROS 2 Humble, ROS-TCP Connector
- Jetson Nano (RT Kernel), Franka ROS 2
- GPU-enabled Workstation 1

4 Hardware Setup and Networking

System Overview

- Workstation 1: Unity, ALVR, SteamVR
- Jetson Nano RT: ROS 2 + libfranka
- Robot: Ethernet to Jetson

IP Configuration (Jetson)

```
network:
  version: 2
  ethernets:
    eth0:
      dhcp4: no
      addresses: [172.16.0.2/24]
```

```
sudo netplan apply
ping 172.16.0.1
```

5 Franka Robot Setup

Franka Desk

- Access via browser at `http://172.16.0.2`
- Control enable/disable
- Error inspection

Installing `libfranka` & `franka_ros2`

```
cd ~/ros2_ws/src
git clone --recursive https://github.com/frankaemika/franka_ros2.git
cd ..
colcon build --cmake-args -DCMAKE_BUILD_TYPE=Release
source install/setup.bash
```

Launch Robot

```
ros2 launch franka_bringup bringup.launch.py robot_ip:=172.16.0.1
```

Real-Time Kernel Check

```
uname -a | grep PREEMPT_RT
```

6 ROS 2 Packages

`franka_cartesian_control`

- Subscribes: `/quest_pose_raw`
- Publishes: `/vr_target`

`franka_vr_control`

- Starts CartesianPoseController
- Publishes at 1 kHz to `ros2_control`

7 Unity Application

```
InputTracking.GetLocalPosition(XRNode.RightHand);
```

8 Networking

- Unity → ROS-TCP-Connector → Jetson
- Jetson publishes to ROS 2 graph

9 Execution Workflow

1. Power on Franka and Jetson
2. Launch bringup
3. Run ROS 2 nodes
4. Launch Unity app

```
ros2 run franka_cartesian_control cartesian_pose_listener
ros2 launch franka_vr_control vr_control.launch.py
```

10 Running Custom Code (`libfranka`)

my_first_motion

Build:

```
mkdir -p ~/franka_ws/my_franka_examples_ws/src
cd src && cp -r my_first_motion .
cd my_first_motion && mkdir build && cd build
cmake .. -DCMAKE_PREFIX_PATH=/home/arpit/franka_ros2_ws/build/
   libfranka
make
```

Run:

```
./my_motion 172.16.0.2
```

11 ALVR + SteamVR Setup on Linux for Meta Quest 2 (USB-C Connection)

This guide explains how to stream VR poses from a Meta Quest 2 headset to your Linux workstation using ALVR and SteamVR, enabling live integration with Unity and ROS 2 for robotic control (as demonstrated in your Franka teleoperation project).

A. Requirements

- A Linux system (e.g., Ubuntu 22.04)
- Meta Quest 2 headset
- USB-C cable (for tethered streaming)
- Steam + SteamVR
- Unity (for app build)
- ALVR (streamer on PC + APK on headset)
- ROS-TCP Connector (Unity $\leftrightarrow$ ROS 2)
- Developer Mode enabled on Meta Quest 2

B. Install ALVR Streamer (Linux)

Option 1: ALVR Launcher (Recommended for beginners)

```
tar -xvzf alvr_launcher_linux.tar.gz
cd alvr_launcher_linux
./ALVR\ Launcher
```

- Click "Add version", keep defaults, and click Install.
- Click "Install APK" to push ALVR app to the headset.
- Launch the streamer app (click Launch).

Option 2: Manual ALVR Streamer (Advanced users)

```
tar -xvzf alvr_streamer_linux.tar.gz
cd alvr_streamer_linux
./bin/alvr_dashboard
```

C. Install and Configure SteamVR

- Install Steam and SteamVR.
- Run SteamVR once and close it (to register drivers).
- In ALVR Launcher → Installation tab → click "Register ALVR driver".
- (Optional) Set ALVR to auto-launch with SteamVR.

D. Connect Meta Quest 2 (USB-C)

- Enable Developer Mode in the Meta Quest mobile app
- Connect Quest via USB-C
- On first connection, accept USB debugging prompt on the headset

E. Install ALVR App on Quest 2

- From ALVR Launcher: click "Install APK"

Alternatively (via SideQuest):

- Install SideQuest
- Connect headset via USB-C
- Drag-and-drop ALVR APK from GitHub into SideQuest to install

F. First Launch and Streaming

- Launch ALVR on PC (streamer)
- Launch ALVR app on Quest headset
- In ALVR streamer → Devices tab → click "Trust" next to Quest
- ALVR begins streaming
- Start SteamVR — if blank, close and relaunch once (to register ALVR driver)

12 Run Instructions: Full System Startup After Setup

Once all IPs and software have been properly configured, follow these instructions to run the complete system.

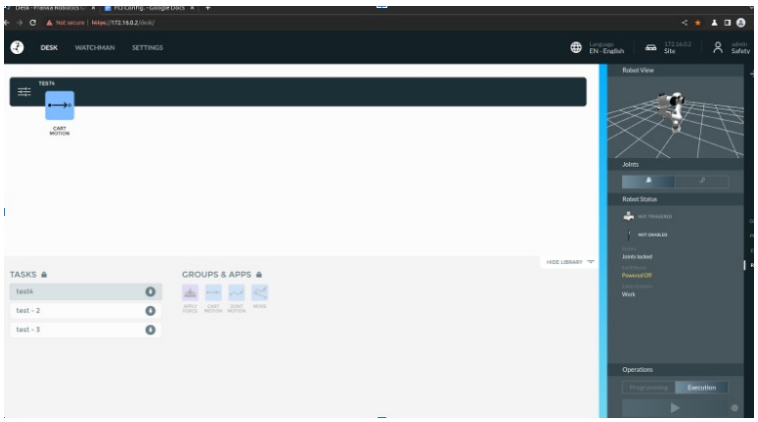


Figure 1: Franka UI interface: unlocking joints and activating FCI

A. Power Up and Open Franka Web UI

Turn on the Jetson Nano and open a browser to `http://172.16.0.2`. If your browser shows a security warning, click **Proceed anyway**.

Unlock the joints and activate FCI after selecting your site from the top-right menu. You should now see a green light on the Franka arm base.

B. Start ALVR + SteamVR on the Workstation PC

Follow the ALVR + SteamVR instructions as described in Section 11.

C. Start ROS TCP Endpoint on Jetson Nano

Open a terminal and source your ROS 2 and franka ROS 2 environment:

```
cd ~/franka_ros2_ws
source /opt/ros/humble/setup.bash
source install/setup.bash
```

Then export the ROS IP (this is Jetson's IP on WiFi network):

```
export ROS_IP=192.168.2.29
```

Now run the ROS TCP endpoint:

```
ros2 run ros_tcp_endpoint default_server_endpoint --ros-args -p
   ROS_IP:=$ROS_IP
```

Expected output:

```
[INFO] [UnityEndpoint]: Starting server on 192.168.2.29:10000
[INFO] [UnityEndpoint]: Connection from 192.168.2.38
```

D. Launch Unity Project on PC

In a second terminal (first terminal is running ALVR), run:

```
unityhub
```

Open the `franka_vr_control` project in Unity 2021.3.3f1 LTS. Ensure that Android Build Support (OpenJDK, SDK/NDK) and Linux IL2CPP support are installed.

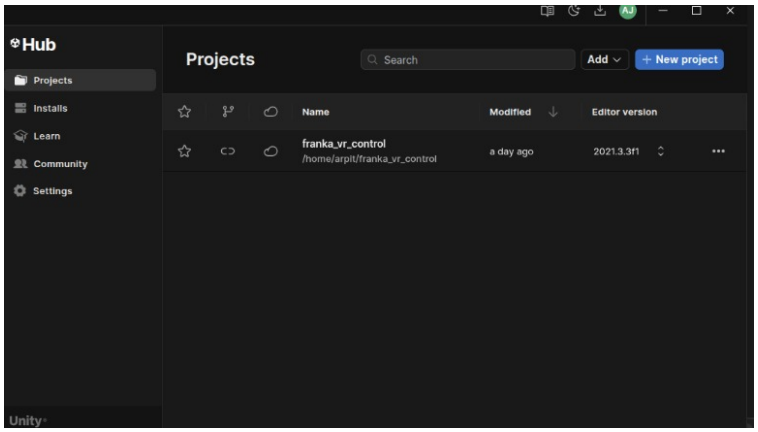


Figure 2: Unity: Opening the franka_vr_control project

Press the `Play` button in Unity. When the connection is successful, Unity and Jetson will exchange messages:

```
[INFO] [UnityEndpoint]: RegisterPublisher(/vr/controller_left/pose,
   ...)
[INFO] [UnityEndpoint]: RegisterPublisher(/vr/controller_right/pose,
    ...)
```

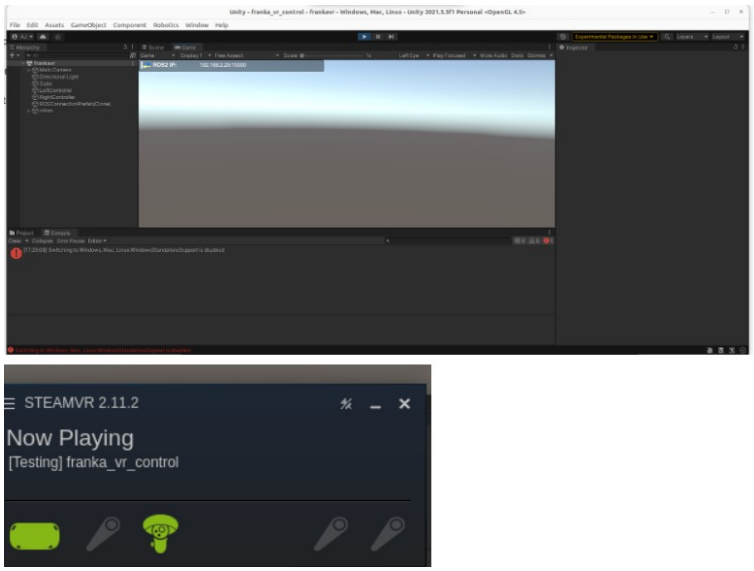


Figure 3: Unity connected and publishing to ROS 2

E. Run Cartesian Pose Listener (Jetson)

Open a second terminal on Jetson, source the same environment and run:

```
ros2 run franka_cartesian_control cartesian_pose_listener
```

Expected output:

```
[INFO] [cartesian_pose_listener]: Streaming to Franka: x=-357.795 y
   =-218.139 z=14.045
```

Keep this terminal open.

F. Calibration: Headset and Controller Placement

Controller aligned below the end-effector for initial pose alignment

G. Launch Final VR Control Node

Open a third terminal on Jetson and run:

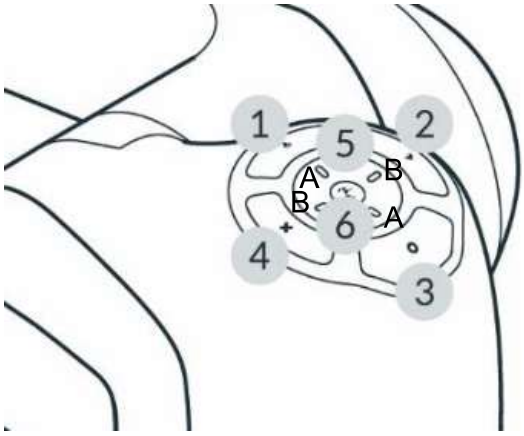
```
ros2 run franka_vr_control vr_cartesian_control 172.16.0.2
```

This starts the Cartesian pose controller and sends the calibrated pose data to the robot. The Franka arm will now move in sync with the Quest controller.

Figure 4: Meta Quest headset placed on Franka base for coordinate calibration

13 Franak UI Instructions

Uses of Buttons on Franka Robot



1. Button no. 1 is for movement of gripper of end effector. First press it once, its icon will be lit with blue light. Now if we double press or long press button 5A then the gripper will close. Similarly if we do that with 6A, then the gripper will open.

2. If button 1 is on and you keep pressing the 5B, the gripper will close gradually. Similarly, if you keep pressing the 6B, the gripper will open gradually.

3. Just switch off the button 1 by pressing it once and the blue light on its icon will turn off.

4. Button 3 can be used to add more poses (If you had opened the move function in your Franka UI, you will be able to see that if you press the button 3 it will add more new poses and if you can navigate to different poses by using button 5B and 6B.

5. If button -2 is illuminated, all changes made in a context menu will be confirmed and skipped to the next section. The button-2 saves any selections made.

6. Delete button (4). If illuminated, pressing the Delete button deletes a selected pose or section.

Buttons - 7 to 9 →

Button - 7

7. To move the robot, half-press the Enabling Button while pressing the Guiding Button simultaneously.

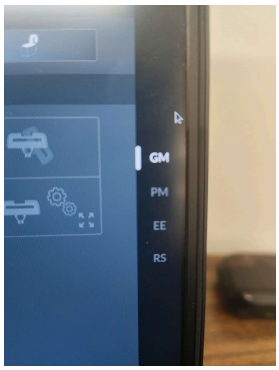
Guiding-Mode Button (8)

8. Button-8 allows the user to change between different guiding modalities by pressing the Guiding-Mode Button.
   The possible Guiding-Modes are translation only, rotation only, free moves, and user-defined movement.

Guiding Button (9)

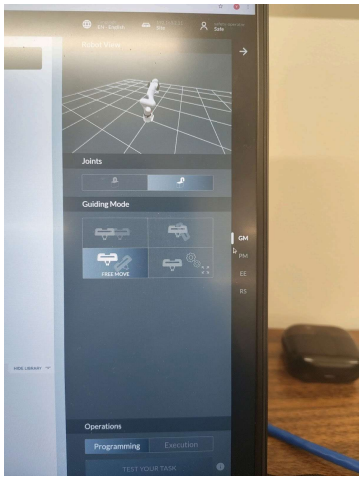
9. The Guiding Button is located on the right of the Pilot-Grip. Press the Guiding Button while half-pressing theEnabling Button (7) simultaneously to move the robot.

FRANKA UI RIGHT SIDE EXPLANATION



We will Talk about these four options in Franka User Interface →

1. GM - Guiding Mode can be used to change the mode of operation of the robot in the UI. We can see Guiding Mode option in our User interface in the picture below -

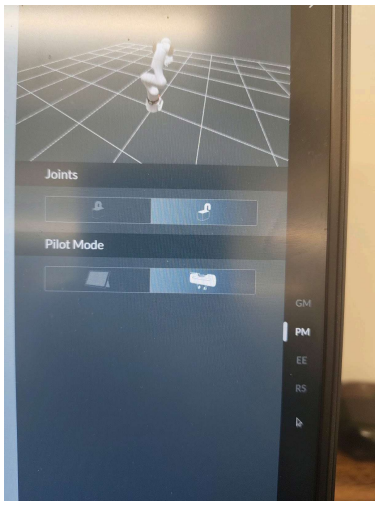


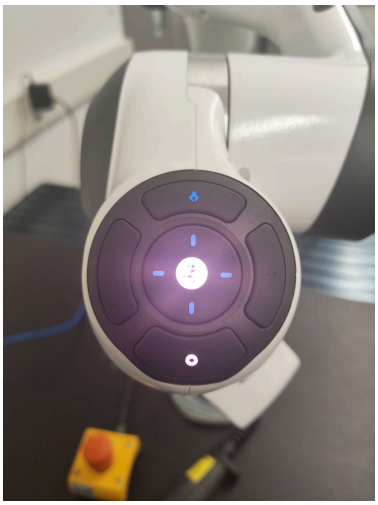
There are four modes of operation-

1. Translation
2. Rotation
3. Free Move
4. User Specified

You can change the mode of operation by using button 8 on the franka robot arm or by using the Guiding Mode (GM) option in the User interface by just clicking on the desired mode.

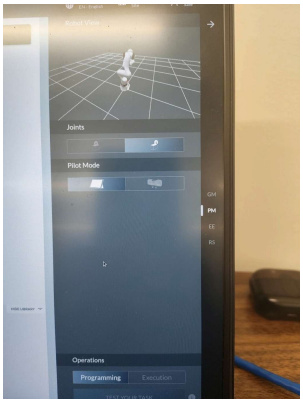
2. PM (Pilot Mode) - It can be used to choose whether you want to control the end effector through the buttons on the franka robot or whether you want to control through your User Interface.





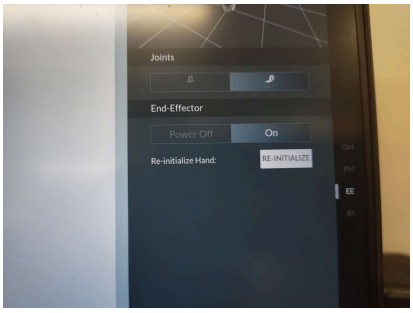
As we can see in above pictures that End effector icon is selected and that's why that button 1 is illuminated with blue light. So now we can use button 5 and 6 for opening and closing of end effector.

If we select the another option having Desk icon →



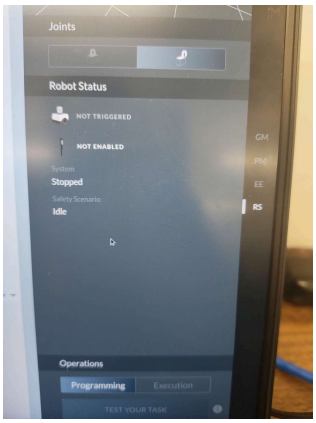
Then the button 1 is no longer illuminated with blue light indicating that you can open or close the effector with the desk (or User Interface) only. So the buttons 5 and 6 will no longer be functional.

3. EE - End Effector →



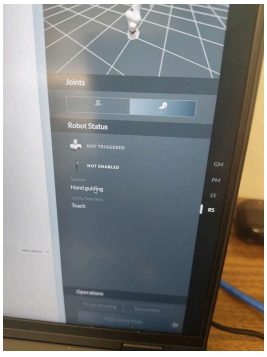
This can be used to power on or power off your End effector.
You can also Re-initialize the end effector by clicking the RE-INITIALIZE option in the UI.

4. RS - Robot Status → It will tell you about your robot current status -



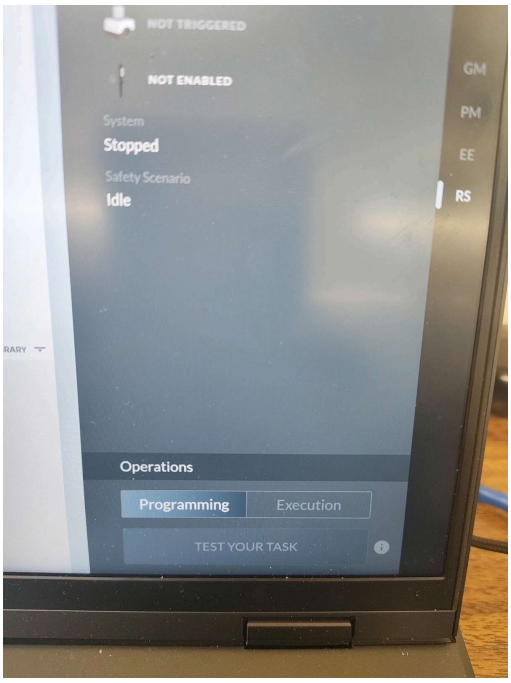
If there is any error, it will be displayed here.

You can see that if you press button 7 and 9 together for free movement of the robot (Hand Guiding mode) , it will be displayed in this robot status also that the system is in Hand Guiding Mode. You can see the picture below →



FRANKA UI RIGHT SIDE EXPLANATION

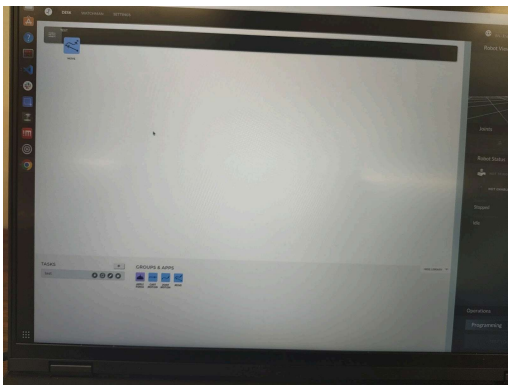
5. Operations Option in the lower right corner of the User Interface→



Here we can switch between the Programming and the Execution mode of the Robot.

In Programming mode we can edit all the things that we want our robot to do when we execute it.

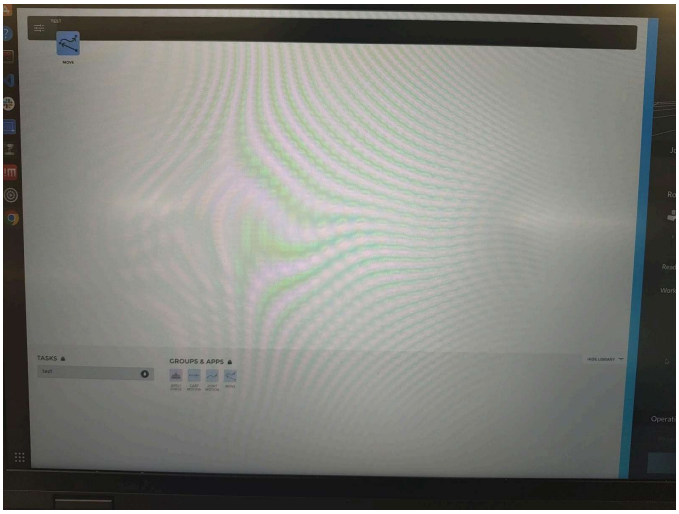
We can see that in the picture below -



FRANKA UI RIGHT SIDE EXPLANATION

Whether in the execution mode we can no longer be able to change the Tasks for our robot.

Here we can see that in the picture below that all the option to edit and addition of tasks are locked-

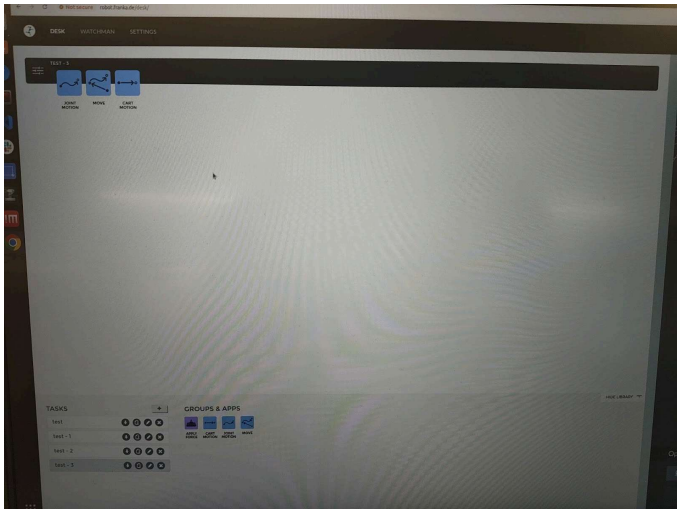


There is one Test you task button also which we can use by keep pressing the X4 key to just test our task on the robot.

We are done with the stuff that is on the right side of the User Interface, Now let's move on to the stuff on the left hand side.

FRANKA UI LEFT SIDE EXPLANATION

1. Here we have this kind of interface-

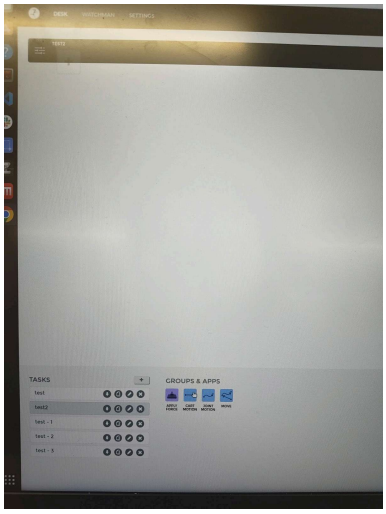


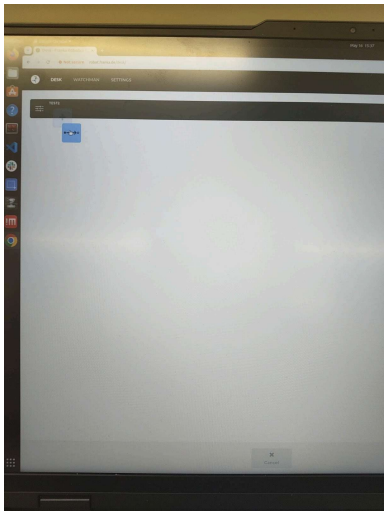
Here we can see that in the tasks section we can create, clone, Rename and delete as many different tasks as we want.

And in the Groups and Apps sections we have 4 apps available-

1. Apply force
2. Cart Motion
3. Joint Motion
4. Move

We can drag and drop anyone we want to use in our task as shown below-



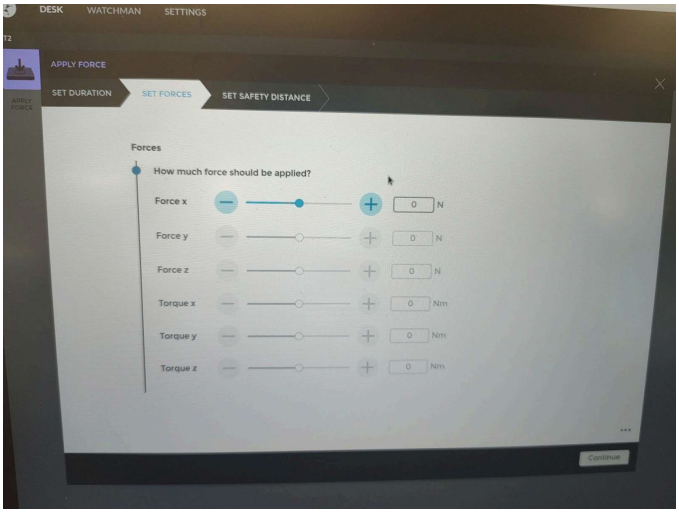


FRANKA UI LEFT SIDE EXPLANATION

We can add as many similar or different kinds of apps as we want in our task.

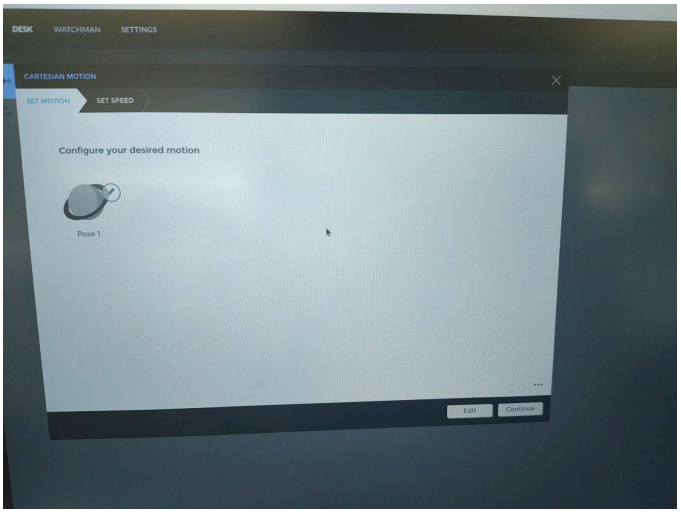
Now let's see different use of different apps available →

1. Apply Force→



There are three options in this which are self explanatory by their name itself.

2. Cartesian Motion→

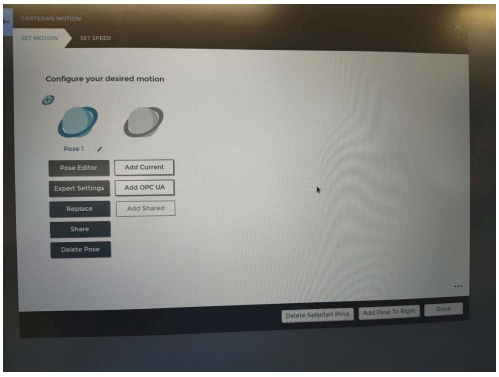


FRANKA UI LEFT SIDE EXPLANATION

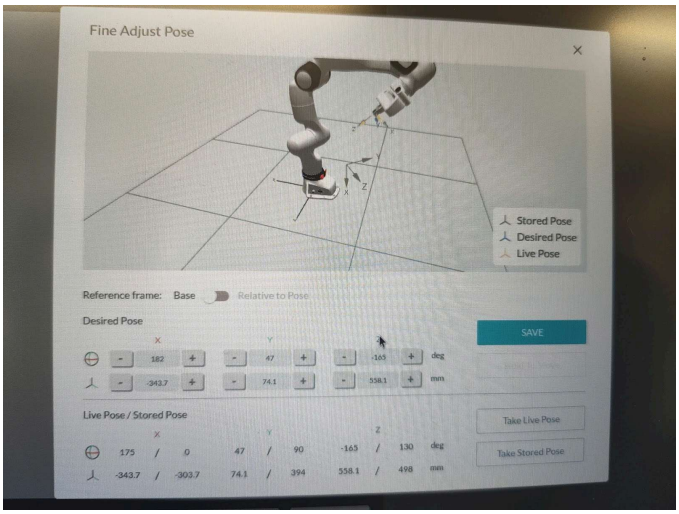
Here we can see that there are two options -

1. Set Motion
2. Set Speed

In Set motion we can add as many different poses as we want to do the desired things.



We can edit each pose also.
Let's talk about editing of the pose -



Here we can see that there are three poses and two reference frame options as Base and Relative to Pose.

FRANKA UI LEFT SIDE EXPLANATION

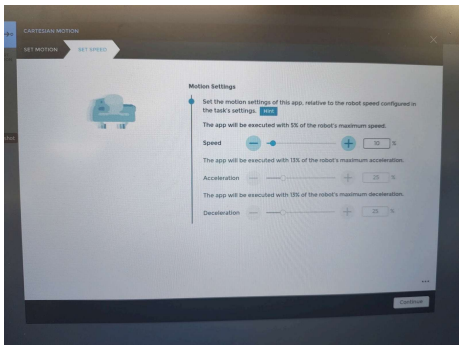
Let's talk about different poses→

Pose Type	Description	When Used	What It Represents
Stored Pose	A **saved pose** (joint or Cartesian) that can be reused	For repeatable tasks, app steps, teaching	Predefined position saved in memory
Desired Pose	The **target pose** the robot is trying to reach	During motion execution	The pose commanded by control/motion planner
Live Pose	The robot's **current real-time pose** from sensors	Always updating during robot operation	Actual measured pose (joint or end-effector)

Let's Talk about the Reference frames→

Reference Frame	Translation Happens From	Rotation Happens From	What It Means / Use Case
Base Frame	Relative to **robot base position**	Relative to **current end-effector orientation**	Use when you want to move to a position fixed in the world, but keep orientation changes local to the tool
Relative to Stored Frame	Relative to **stored pose position**	Relative to **stored pose orientation**	Use when you want to perform motion relative to a saved reference (e.g., move 5cm forward from where the cup was last placed)

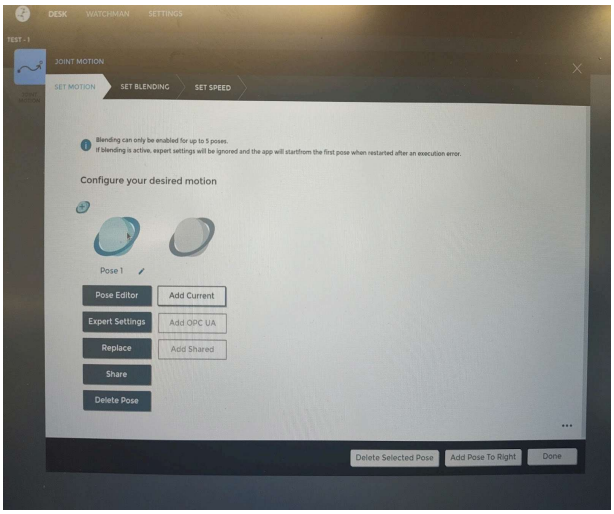
Here we can set speed, acceleration and deceleration of the robot as seen below→

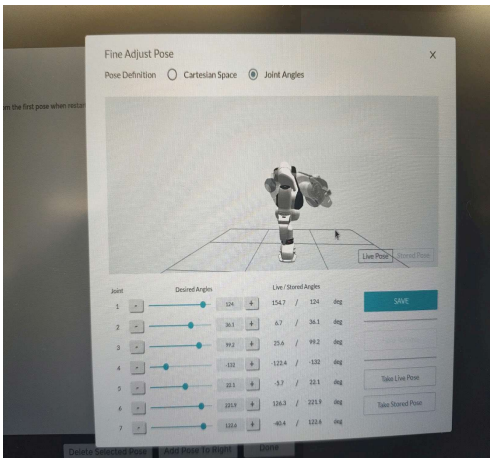


3. Joint Motion

Similarly as cartesian motion we can configure our desired motion by adding different poses in this also.

In the pose editor of this , don't alterate anything in Cartesian space as we are using joint motion here, so only change joint angles to reach desired pose.





We can choose the live pose option and take the stored pose option to initialize our desired pose.

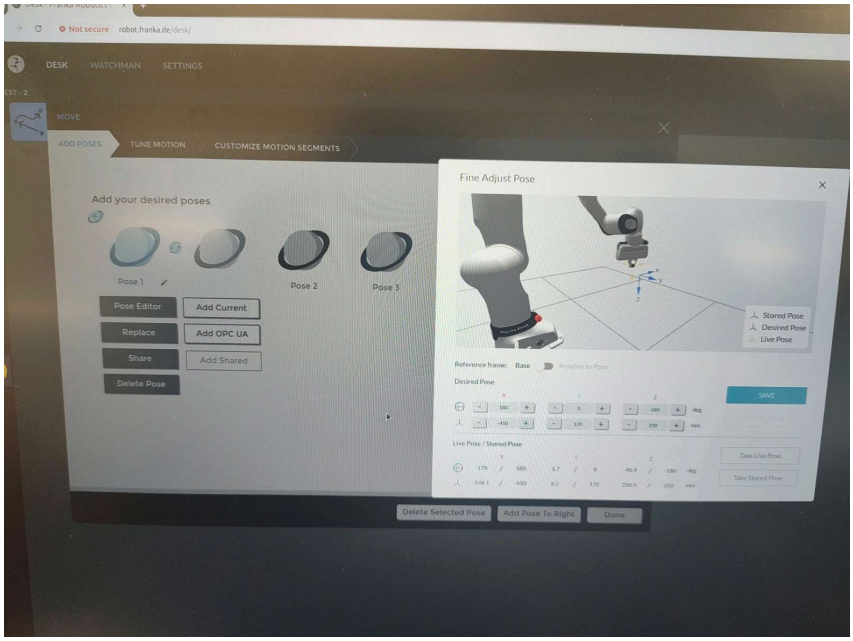
It also has Blending option →

Blending in joint motion allows the robot to **smoothly transition between multiple joint targets** without fully stopping at each one.

It **removes pauses** between motions for **smoother, faster movement.**
Used to make sequences of joint moves feel continuous rather than stop-start.

Then similarly it also has a set speed option which is already defined in the previous app.

4. Move App:



Here we have a similar add poses option as in previous apps.

Here we have tune motion setting also in which we can select motion type and set speed/

Three motion types:

1. Motion to Cartesian Pose:
Moves the robot's end-effector to a specified position and orientation in Cartesian space, following an optimized path that may not be a straight line.

2. Linear Motion to Cartesian Pose:
Moves the end-effector in a straight line to the target Cartesian pose, ensuring a linear trajectory in space.

3. Motion to Joint Configuration:
Moves the robot to a specified set of joint angles, focusing on joint positions rather than the end-effector's path.

In customize motion segments we can customize the individual motion segments of each pose.

Colour	Category	Explanation	Approaching the robot*
White	Not active; ready to start execution	Franka Research 3 is running but not in an active state (either in TEACH or IDLE). It is ready to start execution or interaction.	The robot is inactive and can be approached.
White (slow flashing)	Not active; booting or shutting down	Franka Research 3 is booting or shutting down. Do not interrupt the process.	
White (fast flashing)	Not active; updating	Franka Research 3 is updating. Do not interrupt the process or disconnect Franka Research 3 from its power source.	
Blue	Brakes engaged	Franka Research 3's brakes are locked.	The robot is in a stopped state, but the safety system allows to start motion at any time. Approach with care.
Blue	Ready to execute task	Operator can now start a task in Execution or Test & Jog mode.	
Blue (slow flashing)	Opening brakes	Franka Research 3 is opening its brakes. It might move slowly while doing this.	
Blue (slow flashing)	Collaborative operation (no task active)	Franka Research 3 is in collaborative mode while no task is active.	
Green	Automatic execution	Franka Research 3 is currently running an automatic program and is moving independently.	The robot is executing a task. Do not approach.
Green (slow flashing)	Collaborative operation (task active)	Franka Research 3 is in collaborative mode while a task is active (e.g., to hand-guide in Assist mode).	The robot is executing a task, but ready for collaboration in Assist mode. Approach with care and in accordance with the application-specific safety measures defined for Assist mode.
Green (fast flashing)	Attention: Automatic execution starting	Franka Research 3 will start task execution after "work execution wait time" countdown (if configured) has elapsed.	The robot will soon be executing a task. Do not approach.
Yellow	Warning	Franka Research 3 is in a warning state	The system is in a warning state. Do not approach the robot.
Yellow (slow flashing)	Warning; user interaction needed	Franka Research 3 is in a warning state that needs to be acknowledged by the user.	
Pink (slow flashing)	Conflicting inputs	Franka Research 3 encountered conflicting input signals.	Do not approach the robot.
Red	Error	An error has occurred (e.g., Safety Error, System Error, or loss of communication).	Do not approach the robot.
Red (slow flashing)	Safety Violation / application error	A safety violation or application error is preventing task execution.	The robot can be approached depending on the violation/error shown in the Franka UI and the steps required for recovering a safety violation. Approach with care depending on the situation and actual violation.

14 Troubleshooting

A. Modifying Unity Script for Custom Features

To implement additional control features such as gripper actuation or specific Quest controller button mappings, modify the `XRPosePublisher.cs` script in the `franka_vr_control` Unity project. Once edited, recompile the project and deploy it again through Unity Hub.

B. Frame Alignment and Pose Bias

The system is configured assuming the base of the Franka robot is the origin of the world frame. The Meta Quest headset must be placed at this base during system initialization. The transformation from Quest to robot space is handled in `Cartesian pose listener.cpp`. If a different starting pose is needed (e.g., you reposition the controller or headset), simply adjust the bias variables in that file—there is no need to modify the rotation matrix or main transform logic.

C. Recovery from Invalid Commands or Safety Triggers

If the robot stops moving due to an invalid motion command or safety violation:

- Open the Franka Web UI at `http://172.16.0.2`
- Switch to **Programming Mode**
- Unlock the joints again using the UI interface

Afterward, you can physically move the arm by pressing the two dedicated buttons on the robot. Reference images and instructions are provided in the section detailing the Franka Desk UI.

D. Recovery of Initial Position from a different position

If the robot is in some another position and you want to take it to the initial position :

- Open the build of franka ros 2 and in that open libfranka folder.
- open the example folder
- Run example executables (Preferred: sudo ./generate joint position motion 172.16.0.2)

15 Results and Impact

- Achieved smooth, real-time XR telecontrol on Linux with consumer Meta Quest gear, enabling operators to move the Franka arm as naturally as a handheld tool.
- Demonstrated millimeter-level placement of mock bricks and sensors, indicating readiness for masonry alignment, bolt seating, and structural monitoring tasks.
- Keeps engineers safely off scaffolds and debris while offering an open-source, low-cost platform that construction labs can replicate for training and site automation.

16 Future Work

- Add an eye-in-hand vision module (Intel RealSense) and train a vision-language model so the robot can recognize construction materials and follow spoken commands (e.g., "stack two bricks", "hand me the wrench").
- Develop semi-autonomous task macros—the operator points once in XR and the robot finishes precise placement or tool delivery without continuous joystick-style guidance.
- Integrate onsite progress logging: the robot captures depth images, tags objects, and streams data to BIM databases for real-time construction monitoring.

17 Additional Links

- Internship Poster

18 Acknowledgements

Thanks to the Civil and Environmental Engineering Department, University of Alberta, and to supervisors Yuezhen Gao and Dr. Qipei Mei.