**Setup of UR5e Robot Arm**

1. **ROS2 Humble and Moveit2 ( Ubuntu 22.04 )**

<https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debs.html>

<https://moveit.picknik.ai/humble/doc/tutorials/getting_started/getting_started.html>

1. **UR5e Robot drivers**

<https://github.com/UniversalRobots/Universal_Robots_ROS2_Driver/tree/humble>

<https://docs.ros.org/en/ros2_packages/humble/api/ur_robot_driver/doc/usage.html>

1. **Femto Bolt ROS2 Packages**

[**https://github.com/orbbec/OrbbecSDK\_ROS2**](https://github.com/orbbec/OrbbecSDK_ROS2)

1. **For Calibration**

<https://github.com/giuschio/ros2_handeye_calibration?tab=readme-ov-file>

[**https://github.com/christianrauch/apriltag\_ros**](https://github.com/christianrauch/apriltag_ros)

1. **Python scripting for moveit using ROS2 humble ( Native python scripting is available only from ROS2 Rolling onwards)**

[**https://github.com/AndrejOrsula/pymoveit2**](https://github.com/AndrejOrsula/pymoveit2)

**USAGE and Workflow**

1. **Start up the robot**

ros2 launch ur\_robot\_driver ur\_control.launch.py ur\_type:=ur5e robot\_ip:=192.168.56.101

Start external program on Polyscope

ros2 launch ur\_moveit\_config ur\_moveit.launch.py ur\_type:=ur5e launch\_rviz:=false

You can now control the robot with moveit

/opt/ros/humble/share/ur\_description/urdf/ur\_macro.xacro

To change tool0 location ^

1. **Startup camera and visualize**

ros2 launch orbbec\_camera femto\_bolt.launch.py

Use rviz2 and add respective topics of color image, depth image, ir image or pointcloud to view it in rviz

1. **Calibration - In Progress**

ros2 run apriltag\_ros apriltag\_node --ros-args -r image\_rect:=/camera/color/image\_raw -r camera\_info:=/camera/color/camera\_info --params-file /home/panasonic/ros2\_ws/src/apriltag\_ros/cfg/tags\_36h11.yaml  
  
realsense -

ros2 run apriltag\_ros apriltag\_node --ros-args -r image\_rect:=/camera/camera/color/image\_raw -r camera\_info:=/camera/camera/color/camera\_info --params-file /home/panasonic/ros2\_ws/src/apriltag\_ros/cfg/tags\_36h11.yaml

Above command is for April tag detection. Make sure to change the tags\_36h11.yaml file according to the size of your tag

Then the commands below are for the calibration node and then triggering the service for taking the samples

ros2 launch hand\_eye\_calibration calibration.launch.py

ros2 service call /hand\_eye\_calibration/capture\_point std\_srvs/srv/Trigger {}

This will provide you with the calibration matrix for transforming images in the camera frame to robot base\_link frame

1. **PyMoveit2**

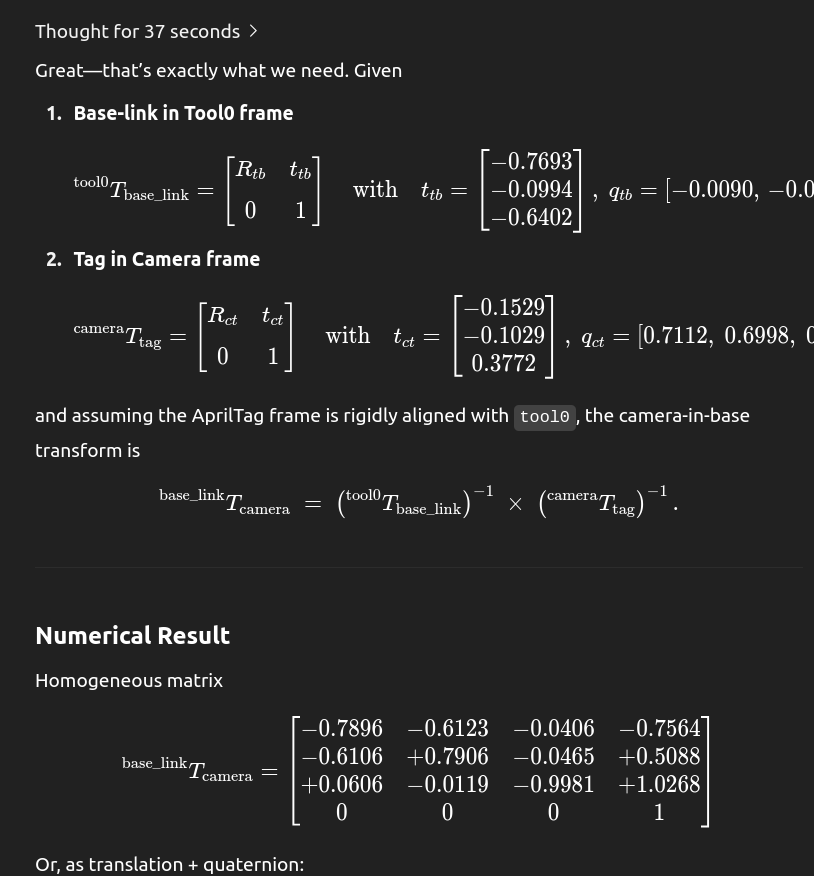
ros2 run pymoveit2 ex\_pose\_goal.py --ros-args -p position:="[-0.8890,0.2984,0.200]" -p quat\_xyzw:="[0.2643,0.9642,0.0161,-0.0137]" -p cartesian:=True

Above is a sample command that can be used for a end effector position that has been found after using the calibration matrix. There are other examples available in the pymoveit2 repo that can be used, but ex\_pose\_goal.py is what is currently being used.

ros2 run tf2\_ros static\_transform\_publisher 0.00 0.00 0.15 0 0 0 1 wrist\_3\_link tool0

ros2 run tf2\_ros tf2\_echo base\_link tool0\_controller

ros2 run tf2\_ros tf2\_echo camera\_color\_optical\_frame tag36h11:0



**June 19th**

1. **Memory -**
2. **Prompts -** Change prompts in prompt file to specify direction in your real world
3. Change env file to ur specific robot

Planners available  
  
BKPIECEkConfigDefault ESTkConfigDefault KPIECEkConfigDefault LBKPIECEkConfigDefault PRMkConfigDefault PRMstarkConfigDefault RRTConnectkConfigDefault RRTkConfigDefault RRTstarkConfigDefault SBLkConfigDefault TRRTkConfigDefault

**Running voxposer envbridge**

ros2 launch ur\_robot\_driver ur\_control.launch.py ur\_type:=ur5e robot\_ip:=192.168.56.101 kinematics\_params\_file:=/home/panasonic/robot\_calibration.yaml

ros2 launch ur\_moveit\_config ur\_moveit.launch.py ur\_type:=ur5e

ros2 launch moveit\_control\_pkg standalone\_tcp\_pose\_server.launch.py ur\_type:=ur5e use\_cartesian:=true

ros2 run tf2\_ros static\_transform\_publisher -0.67450 0.68198 0.95908 0.92211 0.38370 0.03602 0.03441 base\_link camera\_color\_optical\_frame

ros2 run tf2\_ros tf2\_echo base\_link tool0

**ros2 run pymoveit2 voxposer\_moveit.py**

python3 script/ur5\_envbridge\_retry\_savemem\_v1.py 2>&1 | tee log\_envbridge.txt

**Running KUDA**

export PYTHONPATH=$HOME/pyorbbecsdk/build/install/lib:$PYTHONPATH

ros2 run moveit\_control\_pkg tcp\_pose\_server\_kuda --ros-args -p use\_cartesian:=false^C

(kudaoriginal) panasonic@panasonic-All-Series:~/KUDA$ echo $CUDA\_HOME

/home/panasonic/anaconda3/envs/kudaoriginal/pkgs/cuda-toolkit

**CALIBRATION -**

1. Overhead realsense d455

ros2 run tf2\_ros static\_transform\_publisher -0.65929 0.72136 0.85829 -0.37775 0.91128 -0.14943 0.06750 base\_link camera\_color\_optical\_frame

17th July

ros2 run tf2\_ros static\_transform\_publisher -0.49387 0.59460 0.89782 -0.37936 0.92398 -0.04728 0.01018 base\_link camera\_color\_optical\_frame

1. Overhead Orbbec camera

ros2 run tf2\_ros static\_transform\_publisher

-0.61242 0.61839 0.86175

0.91209 0.40958 0.01703 0.00664

base\_link camera\_color\_optical\_frame

1. Wrist camera

ros2 run tf2\_ros static\_transform\_publisher

0.016543 0.089023 -0.163985

-0.004787 0.025668 -0.997618 0.063852

tool0 camera\_color\_optical\_frame

1. Side Realsense camera

ros2 run tf2\_ros static\_transform\_publisher

-0.85868 0.80047 0.29923

-0.33570 0.80193 -0.45527 0.19220

base\_link camera\_color\_optical\_frame

ros2 run tf2\_ros static\_transform\_publisher -0.99265 0.72944 0.35029 -0.35501 0.84646 -0.37527 0.12901 base\_link camera\_color\_optical\_frame

planner\_configs:

SBLkConfigDefault:

type: geometric::SBL

longest\_valid\_segment\_fraction: 0.001

projection\_evaluator: joints(joint\_1,joint\_2)

range: 10

ESTkConfigDefault:

type: geometric::EST

longest\_valid\_segment\_fraction: 0.001

projection\_evaluator: joints(joint\_1,joint\_2)

range: 10

goal\_bias: 0.05

LBKPIECEkConfigDefault:

type: geometric::LBKPIECE

projection\_evaluator: joints(joint\_1,joint\_2)

longest\_valid\_segment\_fraction: 0.001

range: 10

BKPIECEkConfigDefault:

type: geometric::BKPIECE

projection\_evaluator: joints(joint\_1,joint\_2)

longest\_valid\_segment\_fraction: 0.001

range: 10

KPIECEkConfigDefault:

type: geometric::KPIECE

projection\_evaluator: joints(joint\_1,joint\_2)

longest\_valid\_segment\_fraction: 0.001

range: 10

goal\_bias: 0.05

RRTkConfigDefault:

type: geometric::RRT

longest\_valid\_segment\_fraction: 0.001

range: 10

goal\_bias: 0.05

RRTConnectkConfigDefault:

type: geometric::RRTConnect

longest\_valid\_segment\_fraction: 0.001

range: 10

RRTstarkConfigDefault:

type: geometric::RRTstar

longest\_valid\_segment\_fraction: 0.001

range: 10

goal\_bias: 0.05

TRRTkConfigDefault:

type: geometric::TRRT

longest\_valid\_segment\_fraction: 0.01

range: 10

goal\_bias: 0.05

(More available parameters)

PRMkConfigDefault:

type: geometric::PRM

longest\_valid\_segment\_fraction: 0.001

max\_nearest\_neighbors: 100

PRMstarkConfigDefault:

type: geometric::PRMstar

longest\_valid\_segment\_fraction: 0.001

Description of Parameters

-------------------------Parameters Used in all Planners------------------------------------------

longest\_valid\_segment\_fraction =the resolution at which collisions are checked Posible values- From 0.001 to 1 in 0.001 step

-------------------------Parameters Used in Tree-Based Planners------------------------------------------

Range: Represents the maximum length of a motion to be added to the tree of motions. Posible values- From 0 to 10000 in 1 step

-------------------------Parameters Used in Tree-Based Planners which needs a proyection to work ------------------------------------------

projection\_evaluator: Defines the orientation of the projection A good projection in robot arms is the one formed by the shoulder and elbow joints.

-------------------------Parameters Used in Tree-Based Planners with just one tree------------------------------------------

goal\_bias: The process of randomly selecting states in the state space to attempt to go towards, the algorithm may in fact choose the actual goal state, if it knows it, with some probability. This probability is a real number between 0.0 and 1.0 with a 0.01 step

-------------------------Parameters Used in Probabilistic Roadmap Planners PRM------------------------------------------

max\_nearest\_neighbors: represents the maximum number of nodes at which each vertex of the tree can be connected. Posible values- From 8 to 1000 in 1 step

<https://www.traceparts.com/en/search/traceparts-classification-mechanical-components-jig-and-fixture-construction-clamping-clamps-uclamps?CatalogPath=TRACEPARTS%3ATP01007002002008>

<https://www.traceparts.com/en/product/elesa-tru-ushaped-pulling-hooks-for-pulling-hook-clamps-zincplated-steel-ushaped-pulling-hook-trum1046129?CatalogPath=TRACEPARTS%3ATP01007002004006&Product=70-14122020-118298&PartNumber=GG.AX561>

Teleoperation commands

ros2 launch ur\_robot\_driver ur\_control.launch.py ur\_type:=ur5e robot\_ip:=192.168.56.101 kinematics\_params\_file:=/home/panasonic/robot\_calibration.yaml initial\_joint\_controller:=forward\_velocity\_controller

ros2 launch ur\_moveit\_config ur\_moveit.launch.py ur\_type:=ur5e launch\_servo:=true

ros2 service call /servo\_node/start\_servo std\_srvs/srv/Trigger {}

Possibly have to source ws\_moveit first  
  
ros2 launch moveit\_control\_pkg spacenav\_teleop.launch.py

TrackDLO

(hMin = 0 , sMin = 0, vMin = 0), (hMax = 179 , sMax = 255, vMax = 64) - black wire

roslaunch trackdlo realsense\_node.launch

roslaunch trackdlo trackdlo.launch num\_of\_nodes:=25

docker exec -it trackdlo bash

rosrun image\_view image\_saver image:=/camera/color/image\_raw