# Solving Tower of Hanoi with Vision

## xArm6 Robot with RGBD camera, ROS2, Gazebo & MoveIt2

Shankho Boron Ghosh

Smart Robotics, UniMoRe

January 2026

# Problem Statement

**Tower of Hanoi Puzzle:**

- Classic mathematical puzzle
- 3 pegs (bins), 3 disks (aruco boxes) of different sizes
- Move all disks from source to target
- **Rules:** Only move top disk, never place larger on smaller

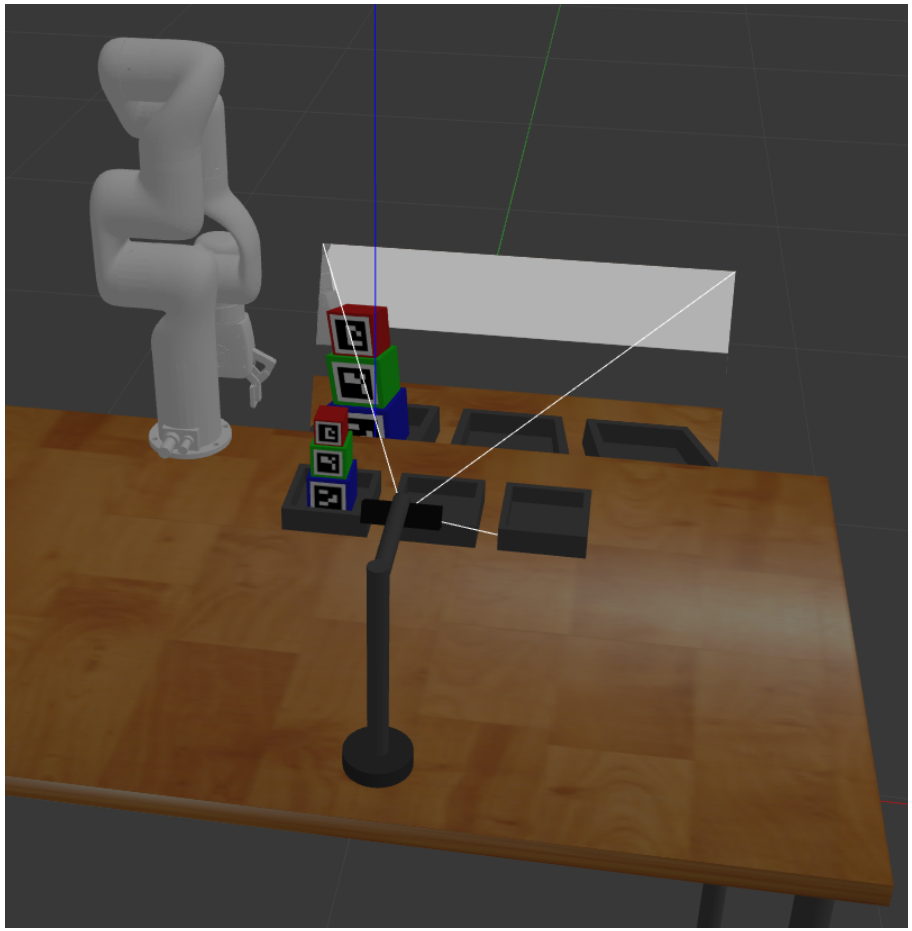
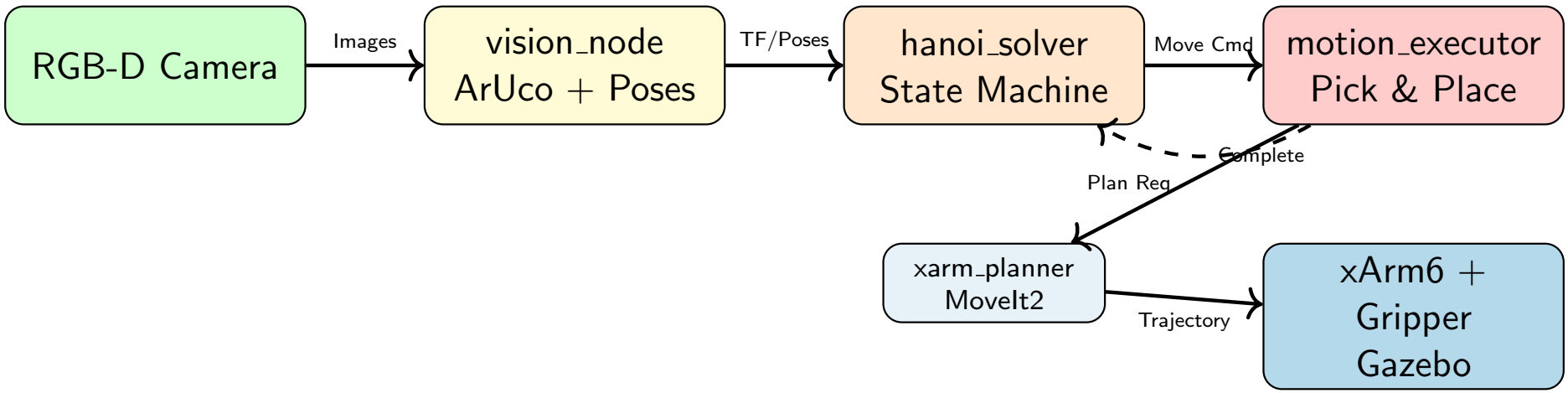


Figure: Gazebo simulation environment

# System Architecture Overview




**ROS2 Nodes:**

- `vision_node.py`
- `hanoi_solver.py`
- `motion_executor.py`

**External Packages:**

- `xarm_ros2` (UFACTORY)
- MoveIt2 + OMPL
- Gazebo Classic

# ROS2 Node Graph

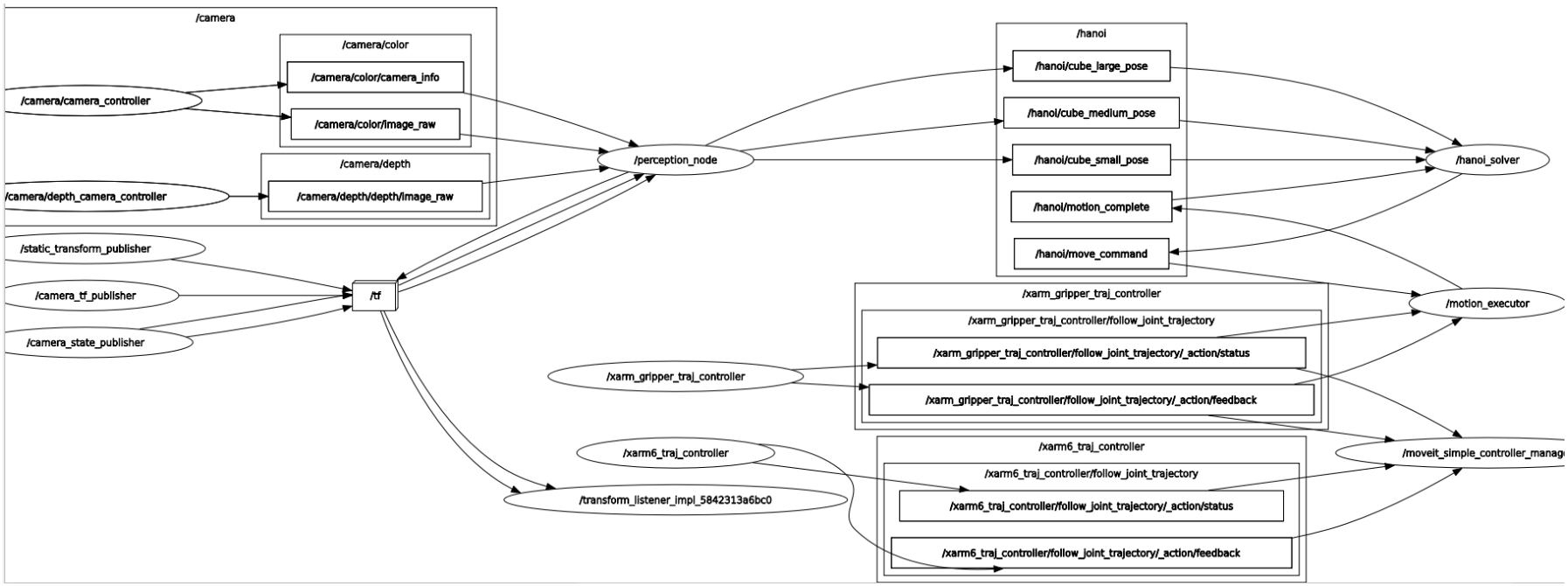



Figure: `rqt_graph` showing relevant node connections and topics

# xArm6 Robot & Gripper

## UFACTORY xArm6:

- 6 DoF Anthropomorphic Robot
- 700mm reach
- 5kg payload

## UFACTORY xArm Gripper:

- Parallel jaw mechanism
- `drive_joint`: 0.0 (open) → 0.85 (closed)
- 172mm TCP offset
  link_base→...→link6→link_eef(xarm_base_link)→link_tcp

## RGB-D Camera:

- Standard Gazebo
- Eye-to-hand configuration (External Camera)
- Mounted facing the taskspace

Figure: xArm6 with gripper

# Scene Layout

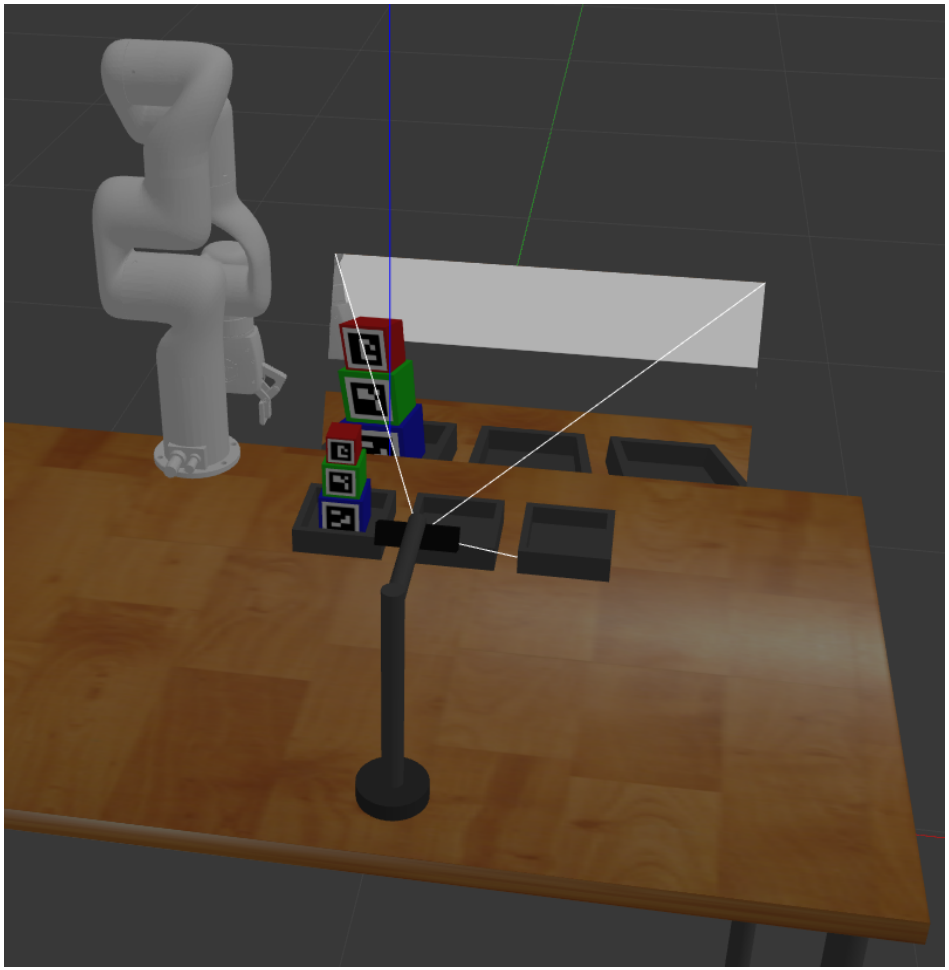


Figure: Gazebo simulation scene

**Components:**

- **Robot**
- **Camera with Stand**
- **3 Bins:** A, B, C (left to right)
- **3 Cubes:**
  - Small (40mm, Red, ArUco 1)
  - Med. (50mm, Green, ArUco 2)
  - Large (60mm, Blue, ArUco 3)
- **Table:** Surface at z=1.015m

**Initial State:**

All cubes stacked on Bin A
(Large $\rightarrow$ Medium $\rightarrow$ Small)

# Vision Node: Cube Pose

## ArUco Detection Configuration:

- Dictionary: `DICT_4X4_50`
- Corner refinement: `SUBPIX`
- Processing rate: 10Hz

## Pose Estimation:

- `solvePnP_IPPE_SQUARE`
- Optimized for square markers
- Computes rotation + translation

## Depth Fusion:

- Sample 5×5 ROI at marker center
- Filter: $0.1\text{m} \leq \textit{depth} \leq 3.0m$
- Take median of valid samples

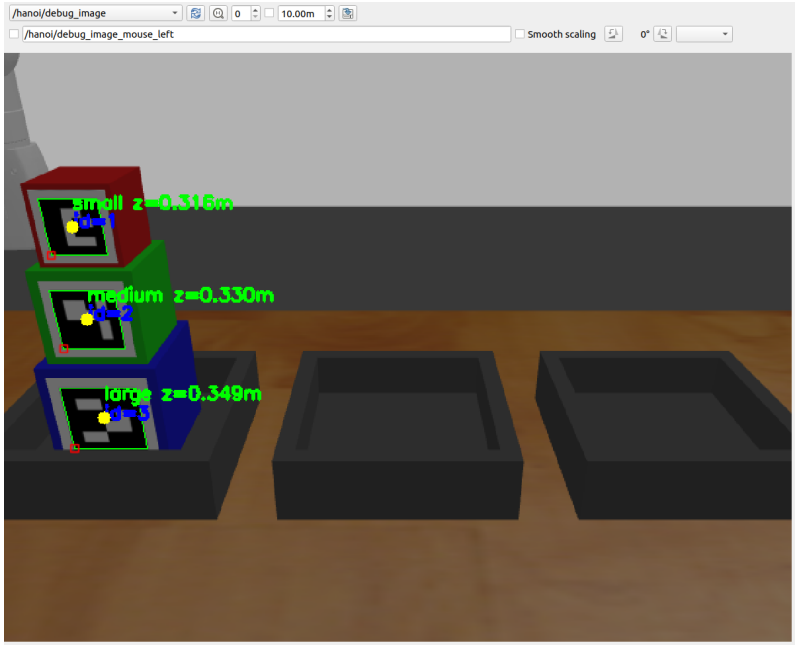
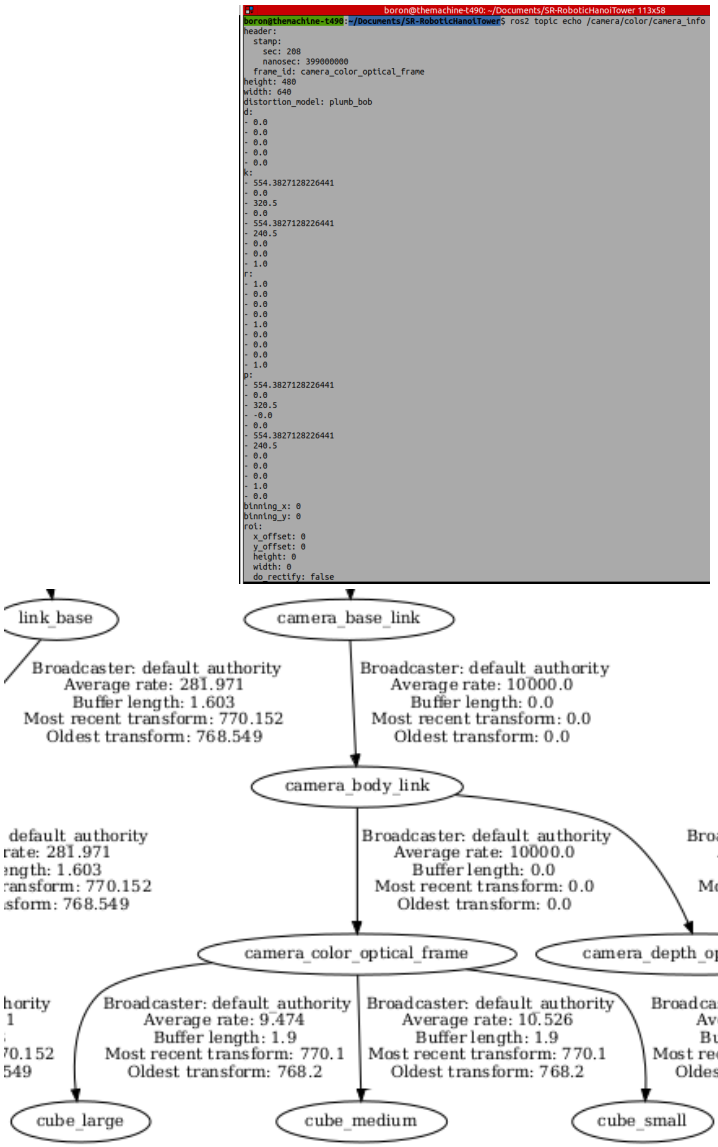



Figure: ArUco detection

**Topic:** `/hanoi/debug_image`

# Vision Node: TF Frame




## 3D Position (Pinhole / Plumb Bob):

$$X = \frac{(u - c_x) \cdot Z}{f_x}$$

$$Y = \frac{(v - c_y) \cdot Z}{f_y}$$

$$Z = \text{depth (from sensor)}$$

| | |
|---|---|
| $(u, v)$ | pixel coordinates |
| $(f_x, f_y)$ | focal lengths |
| $(c_x, c_y)$ | principal point |

## Cube Center Offset:

Marker on front face $\rightarrow$ offset by `cube_depth/2` along marker Z-axis

# Hanoi Solver Node

## Solving using recursion:

```
def hanoi(n, src, tgt, aux):
    if n == 0: return
    hanoi(n-1, src, aux, tgt)
    move(disk_n, src, tgt)
    hanoi(n-1, aux, tgt, src)
```

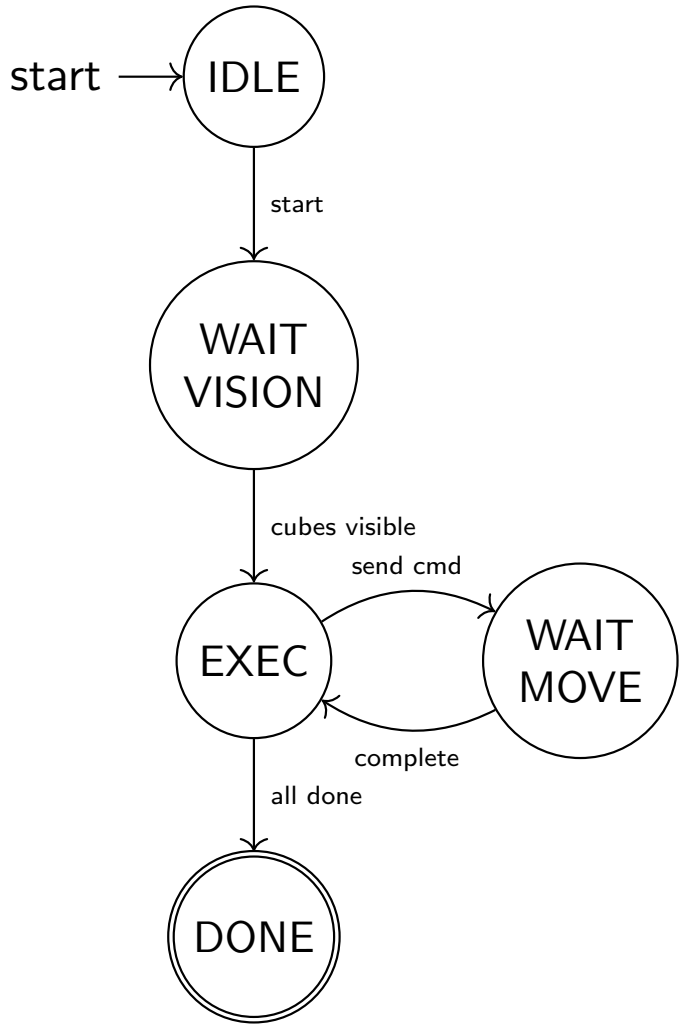
## Optimal Moves: $2^n - 1$

For 3 disks: **7 moves**

## Move Sequence:

1. small: A → C
2. medium: A → B
3. small: C → B
4. large: A → C
5. small: B → A
6. medium: B → C
7. small: A → C

## Solver State Machine:



# MoveIt2 + xarm_planner Integration

**Motion Planning:**

- OMPL motion planning library

**RRTConnect** (Rapidly-exploring Random Tree Connect)

- Bidirectional sampling-based planner
- Grows trees from start & goal

**xarm_planner Services:**

| Service | Purpose |
|---|---|
| `/xarm_pose_plan` | Cartesian pose (IK+RRT) |
| `/xarm_straight_plan` | Cartesian (IK, 5mm steps) |
| `/xarm_exec_plan` | Execute trajectory |

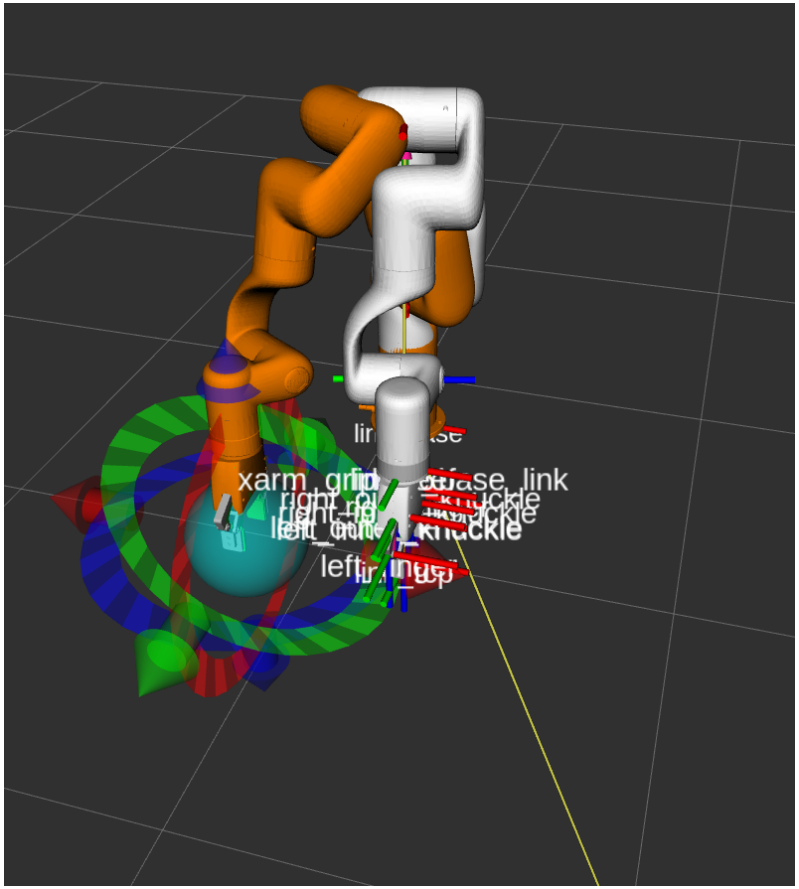
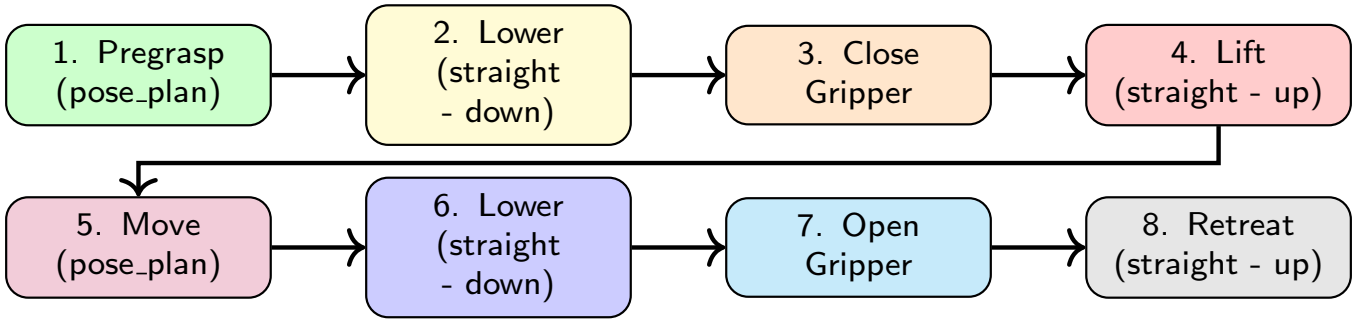


Figure: Motion planning in RViz

**Control Configuration:** Centralized position control (with velocity feedforward) @ 150Hz

# Motion Executor Node: Pick-and-Place Sequence




**Free-space motion:**

`/xarm_pose_plan` (RRTConnect)

- Steps 1, 5

**Cartesian motion:**

`/xarm_straight_plan` (5mm steps)

- Steps 2, 4, 6, 8

# Grasping Emulation: IFRA_LinkAttacher

**Problem:**

Gazebo gripper contact physics unreliable, cubes tend to "slip out / fly out" during motion.

**Solution:**

`IFRA_LinkAttacher` Gazebo plugin creates rigid joint on grasp:

| Service | Type | Action |
|---|---|---|
| /ATTACHLINK | AttachLink | Fixed joint |
| /DETACHLINK | DetachLink | Remove joint |

**Service Request Fields:**

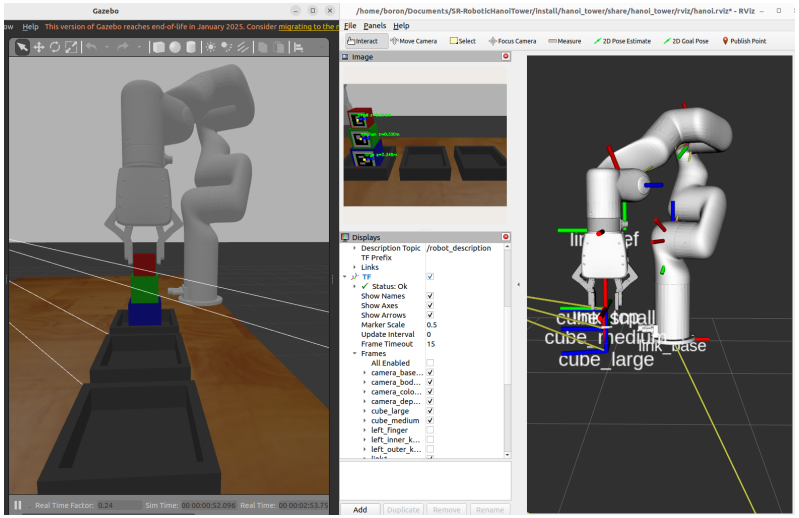
```
req.model1_name = 'UF_ROBOT'
req.link1_name  = 'link6'
req.model2_name = 'cube_{name}'
req.link2_name  = 'cube_link'
```

**Adaptive Gripper Position:**

```
# Cube sizes (m)
small, medium, large = 0.04, 0.05,
      0.06

grip_inwards = 0.006   # squeeze 6mm
max_gap = 0.085        # gripper max
max_pos = 0.85         # joint limit

closing = cube_size - grip_inwards
pos = max_pos * (1 - closing/max_gap)
```



# ROS2 Topics & Services

**Published Topics:**

| Topic | Type |
|---|---|
| /hanoi/cube_*_pose | PoseStamped |
| /hanoi/debug_image | Image |
| /hanoi/solver_state | String |
| /hanoi/motion_state | String |
| /hanoi/current_move | String |
| /hanoi/move_command | String |
| /hanoi/motion_complete | Bool |

**Gripper Control:**

| Joint | `drive_joint` |
|---|---|
| Open | 0.0 |
| Closed | 0.85 |

via FollowJointTrajectory action

**Services:**

| Service | Type |
|---|---|
| /hanoi/start | Trigger |
| /hanoi/reset | Trigger |
| /hanoi/pause | SetBool |
| /xarm_pose_plan | PlanPose |
| /xarm_straight_plan | PlanSingleStraight |
| /xarm_exec_plan | PlanExec |
| /ATTACHLINK | AttachLink |
| /DETACHLINK | DetachLink |

# Simulation Demo

## Initial State

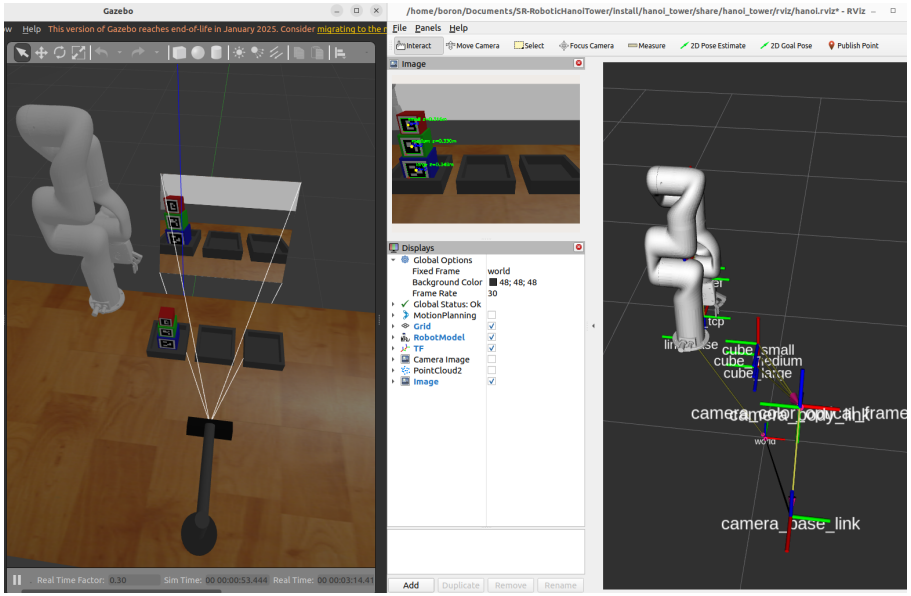



Figure: Cubes on Bin A

## Final State

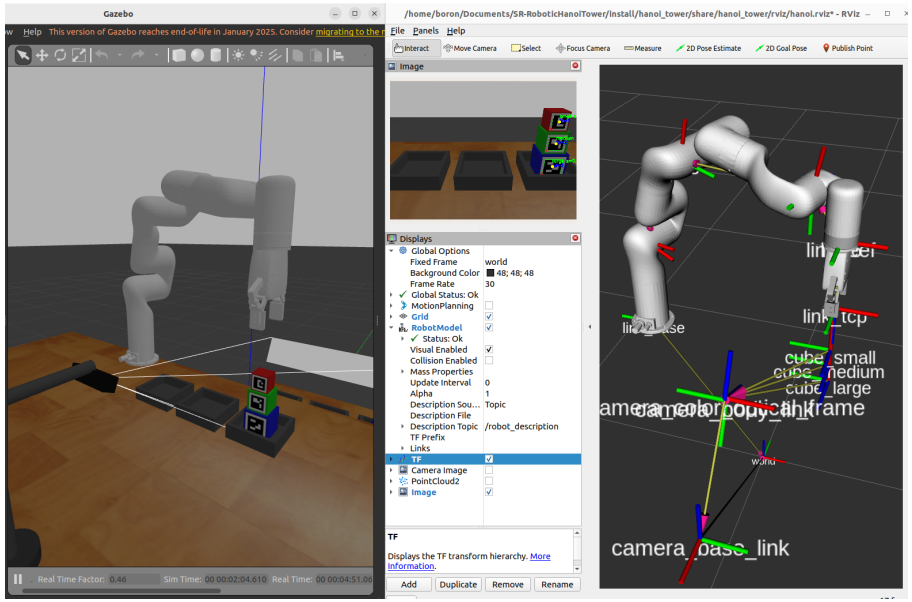



Figure: Cubes on Bin C

## Commands:

```
ros2 launch hanoi_tower demo.launch.py
ros2 service call /hanoi/start std_srvs/srv/Trigger
```

# Challenges

| **Challenge** | **Solution** |
|---|---|
| Cube physics instability | Increased friction, spawn delays (10s between cubes) |
| TF frame accuracy | Multi-point depth sampling, solvePnP refinement, marker-to-cube-center offset |
| Gripper grasp reliability | IFRA_LinkAttacher for rigid joint attachment in Gazebo |
| Real-time factor delays | Adjusted timing parameters (42s+ startup delay for 0.3 RTF) |
| Service timeout issues | Non-blocking with timeout handling |

# Conclusion

**Future Work:**

- Variable number of disks
- Dynamic obstacle avoidance
- Deep learning for perception
- Hand-eye calibration
- Hand in Eye (Camera on EE)
- Real robot deployment

# Thank You!

Questions?

 github.com/sboronghosh/SR-RoboticHanoiTower