Job Assignment Robotics Engineer

Tools & Frameworks (brief):
We use **ROS 2** for its robust, real-time middleware; **MoveIt** for ready-made IK/FK solvers and motion planning; and **RViz** to visualize robot state, waypoints, and trajectories.

Task 1: Robot-Arm Setup

- **Robot:** UR5 (6 DOF)
- **Implementation:**
    1. Load the UR5 MoveIt configuration package.
    2. Define each joint's min/max limits (already provided by the URDF).
    3. Use MoveIt's /compute_fk and /compute_ik services to support forward and inverse kinematics.

Task 2: User Input

- **Interface:** Read six floats (x, y, z, roll, pitch, yaw) from the command line.
- **Validation:**
    1. Call MoveIt's IK service (/compute_ik) to ensure the Cartesian target is reachable.
    2. Reject or warn if the pose is outside joint or workspace limits.
- **State Tracking:** Store the current end-effector pose (via FK) for interpolation.

Task 3: Smooth Movement

- **Method:** Linear interpolation in Cartesian space over a fixed number of steps .
- **Process:**
    1. Compute intermediate poses by blending start and goal vectors.
    2. For each waypoint, call IK to get joint targets.
    3. Append these to a JointTrajectory with appropriate timestamps.

Task 4: Execution & Visualization

- **Execution:** Send the completed JointTrajectory to the /ur5_arm_controller/follow_joint_trajectory action server and monitor its result.
- **Visualization:**
    - Publish each Cartesian waypoint as an RViz Marker (spheres) on the waypoint_markers topic.
    - Observe the planned path and execution in RViz.
- **Logging:** Print each waypoint's Cartesian coordinates to the console for verification.

As shown in Figure 1, the visualization of the computed Cartesian trajectory is displayed in RViz, alongside the logging output that documents each waypoint and the execution status in the terminal.

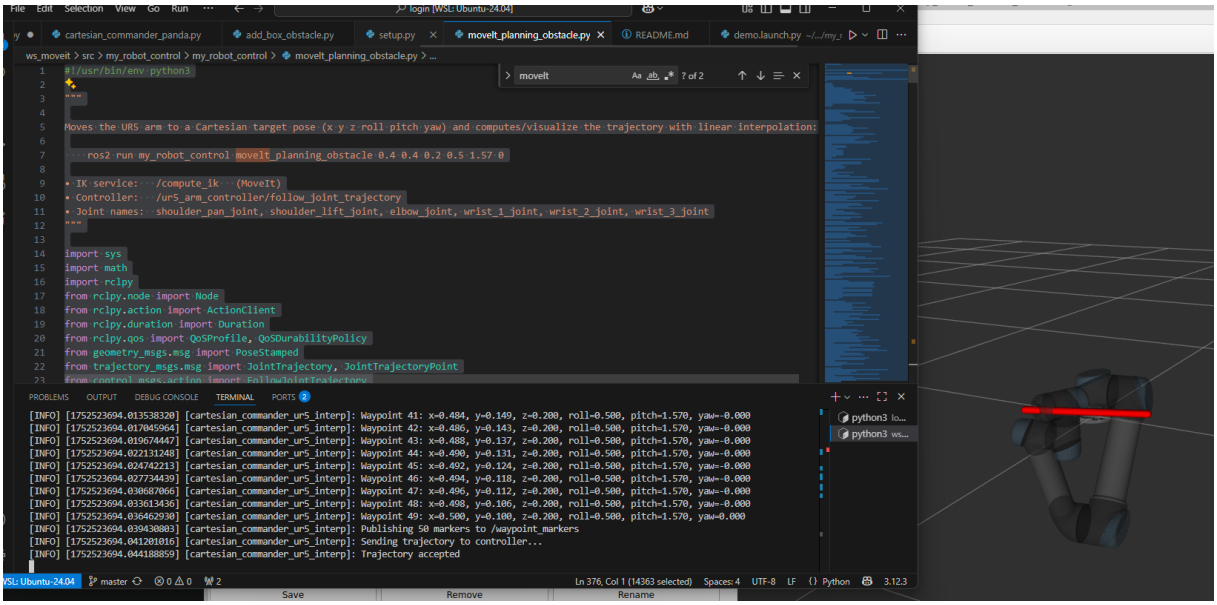

Figure 1 Screenshot Visualization of the trajectory and logging of the steps