ELECTRICAL TEAM TRAINING

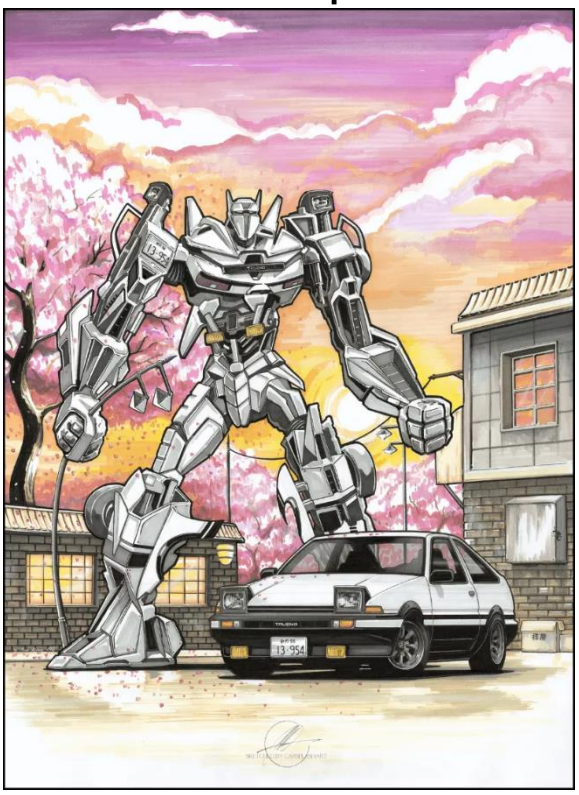
TASK 12

Individual Tasks

Task 12.1 Build Your Own Robot

Preface:

A new bot has just joined the Autobots and has chosen the Toyota AE86 as his transformation vehicle. Help him simulate the transformation and explore how he can add sensors to detect malicious Decepticons.



Requirements

Simulate a 4-wheel robot in RViz using URDF/XACRO.

Your robot must include:

- Base Link (main body of the robot).
- Four Wheels (two on the left, two on the right).
- LiDAR Sensor mounted on the base link.
- Proper links and joints connecting all components.

Deliverables

1. XACRO File
    - Define the base link, wheels, and LiDAR sensor.
    - Use appropriate link and joint tags to connect them.
    - Make sure the wheels are positioned correctly relative to the base.
    - Include simple shapes (boxes/cylinders) for visualization.
2. Launch File
    - Launch Gazebo
    - Set the robot_description parameter in the ROS parameter server (Review Session #2)
    - Launch the necessary packages to load the parsed URDF file
    - Launch a pre-configured RViz file to show the robot model and TF tree.

Task 12.2 Transform your troops

Preface

Malicious Decepticons have used their **aerial** abilities to strike our troops on a new **battlefield**. We must level up and **relocate** our units to new coordinates. Use transformations to move our troops to the new war zones.

Requirements

Write a Python script or Jupyter Notebook that uses Homogeneous Transformation Matrices (HTMs) and quaternion transformations to move your troops in a 3D battlefield.

Deliverables:

- A Python Script / Jupyter notebook.
    - Take an array of points i.e. (Point cloud) as input
      [[x, y, z], [x, y,z], ..]
    - Take the required translation in all three axes (t_x, t_y, t_z)
    - Take the required rotation about all three axes [θ_x, θ_y, θ_z]
    - Choose whether to use HTMs or a Quaternion-based approach to calculate the resultant points.
    -
    - Plot the original and transformed points in a 3D cartesian plane using the matplotlib library.

Group Task 12.3:
Extend and Control Your Robot

Preface

The Autobots are upgrading their capabilities for the battles ahead. Your mission is to enhance your robot with new mechanical and computational features, enabling it to adapt and fight more intelligently against the Decepticons.

Requirements

1. Modify the Robot Model
    - You will be given the Shato robot .xacro files OR use your own .xacro file(s).
    - Modify them to add a 2-DOF (Degree of Freedom) robotic arm.

2. Custom Robot State Publisher
    - Create a Python node that replicates the core functionality of robot_state_publisher for your robot.
    - The node should:
        - Subscribe to a joint state topic (e.g., /joint_states).
        - Use the received joint angles and the hardcoded dimensions of your robot arm (link lengths, offsets) to calculate the transformation matrix for each link.
        - Publish these dynamic transforms to the /tf topic for visualization in RViz.

Note: Do not parse the URDF file directly. Hardcoding the dimensions is the intention to not waste time parsing XML

3. Simulation Launch File
    - Create a launch file that:
        - Starts the Gazebo simulation environment with your complete robot.
        - Launches RViz with a pre-configured .rviz file showing the robot model, TF frames, and more.
        - Starts your custom nodes for publishing to /tf and any other required nodes.
    - Includes a launch argument named use_lidar (true/false) that conditionally adds a LIDAR sensor model in the robot's URDF.

4. (Bonus) Arm Control Node
   - Implement a second Python node that acts as the arm's "brain."
   - This node should:
     - Subscribe to the /move_base_simple/goal topic to receive target poses from RViz's 2D Nav Goal tool.
     - Implement inverse kinematics equations to calculate the required joint_1 and joint_2 angles for the target (x, y, z) coordinate.
     - Detect if a goal is unreachable; if so, print a warning and take no action.
     - If the goal is reachable, publish the calculated joint angles to a custom topic (e.g., /joint_commands).
     - Update your Task 2 node to listen to this new topic.

5. (Bonus) Omni/Mecanum Wheel Kinematics
   - Create a separate node that implements mobile base kinematics for an omni or mecanum wheeled robot.

   - This node should:
     - Subscribe to the /cmd_vel topic (using geometry_msgs/Twist messages).

     - Use the incoming linear (x, y) and angular (z) velocities to compute the required rotational velocities for each of the four individual wheel joints.

Best of Luck! :3