Dynamic Object Sorting & Placement Using Grasping

By - Atharva Kulkarni, Gayatri Mangire, Aryan Solanki

Objective

Goal: Develop a system for autonomous object sorting and placement.

Focus: Realistic simulation, robust vision processing, and efficient robot operations.

Environment Setup

Simulation in Gazebo:

- *Launch File Features:*
    - Configures Gazebo model paths and environment variables.
    - Includes arguments for customizing world setup and robot behavior.
    - Loads the Gazebo world and integrates the Fetch robot with flexibility for various needs.

Defining Models:

- Models structured using `model.config` (metadata) and `model.sdf` (links, joints, poses).
- Models used: Box (capacity labels), Cafe Table, and Cubes.

World Configuration:

- Specifies layout, ground, lighting, and interactive objects in the virtual environment.
- Configurations inspired by **SceneReplica Paper (IRVL UTD)** for cube placement.

Key Components:

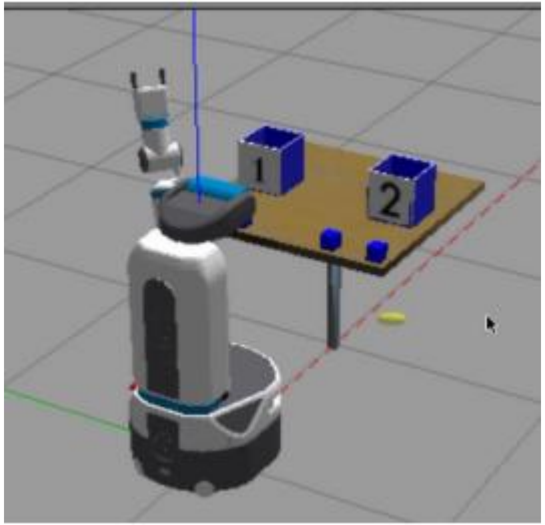
    - Box, cube, and table models with precise physical properties.
    - Uniform lighting and strategic robot placement.

Box Capacity Detection

- **Key Steps:**
  - **Image Acquisition:** High-resolution images via Fetch robot's camera.
  - **Preprocessing:** Grayscale conversion, thresholding, and ROI cropping using OpenCV.
  - **OCR with Tesseract:** Extract numerical capacities

    placement logic.
- **Output:** Centralized dictionary of box capacities

  for decision-making.



Grasp Planning

Planning for Object Interaction:

- Uses TRAC-IK for inverse kinematics to compute joint trajectories.
- Adjusts rotation and translation matrices for accurate grasping (e.g., `rotY`, `ros_pose_to_rt`).
- Computes pose transformations for precise object handling.

Pre-Grasp Positioning:

- Calculates safe pre-grasp positions using offset translations.
- Positions the arm with **MoveGroupInterface** for effective grasping.

Collision Detection:

- **Ensuring Safety:** Defines objects with `CollisionObject` messages.
- Configures object dimensions and poses for collision-free motion.
- Publishes collision objects to `/collision_object` topic, ensuring accuracy through publisher delays.

Robot Motion Control

Motion Planning Techniques:

- **move_group.moveToPose:**
  - Moves robot to a target pose in Cartesian space, defined by position and orientation.
- **move_group.moveToJointPosition:**
  - Directly sets target joint angles for precise arm configurations.

Why Use MoveGroupInterface?

- Designed for modern ROS workflows with active support.
- Offers streamlined and optimized APIs for efficient motion planning compared to older alternatives like MoveGroupCommander.

Execution Features:

- Provides smoother trajectories and collision-free movements.
- Ensures reliable and repeatable robot operations.

Key Results

- **Achievements:**
  - Accurate OCR-based capacity detection.
  - Stable grasping and object handling.
  - Efficient sorting and placement in dynamic scenarios.

Conclusion & Future Work

- **Conclusion:**
  - Demonstrated a robust and scalable robotic sorting system.
- **Future Scope:**
  - Integration in real-world industrial setups.
  - Enhancements in dynamic adaptability and clutter handling.