Robotics Using Ned2

Team Members:

1. PES1UG20CS435 - Sriram R
2. PES2UG20CS153 – Kamalesh S

Title: Automated Refuelling for Automated Vehicles/Satellites for (ARAV/S)

Abstract:

This abstract describes a monolithic testing approach to evaluate the extensive functionality of the Niryo robot in automatically refuelling autonomous vehicles. The code implementation leverages the niryo\_robot\_python\_ros\_wrapper library and the Robot Operating System (ROS) to perform various actions, including calibration, object detection, and manipulation using the Niryo robot. The primary objective is to demonstrate the robot's capabilities in identifying and manipulating red circular objects within a designated workspace.

Introduction:

The Niryo robot is an advanced robotic system capable of various tasks, including object detection, manipulation, and interaction with the environment. This extended abstract presents a Python-based monolithic testing script that showcases the robot's capabilities in identifying and handling specific objects within a predefined workspace. The program focuses on detecting the vehicle’s fuel tank and verifying the successful refuelling using the robot's vacuum pump, in this project we will represent the vehicle’s fuel tank as a red circular object and use the robot’s vacuum pump to represent the fuel pump.

ROS and Niryo ROS Wrapper Initialization:

The code begins by initializing the ROS node, which allows communication with the Niryo robot system. It utilizes the niryo\_robot\_python\_ros\_wrapper library to create a connection to the robot hardware and control its actions.

Observation Pose:

The observation pose is a predefined configuration in 3D space that serves as the reference position for the robot during object detection and manipulation. The pose is represented by a tuple of six values (x, y, z, roll, pitch, yaw).

Calibration:

Before executing any task, the robot undergoes automatic calibration. Calibration is crucial to ensure accurate and precise movements during the object manipulation process.

Vacuum Pump Initialization:

The Niryo robot is equipped with a vacuum pump, which is essential for picking up and releasing objects, but here, we will use it to represent a fuel pump. The code initializes the vacuum pump to create suction required for object manipulation.

Object Detection and Manipulation Loop:

The core part of the program consists of a loop that runs twice (i.e., for two iterations) to detect and manipulate the fuel tank in the specified workspace. The loop is structured as follows:

a. Object Detection: The robot performs a vision-based object detection task to find objects in the designated workspace. The vision\_pick function is used for this purpose, taking parameters such as workspace\_name, height\_offset, shape, and color. In this case, the code searches for objects of any shape and any color, but the desired object is a red circular one.

b. Verification and Manipulation: Upon detecting an object, the code checks if the object is red and circular. If the criteria are met, the robot proceeds with the manipulation process. It moves to the detected object's position, using the move\_to\_object function with specified parameters such as workspace, height\_offset, shape, and color.

c. Vacuum Pump Activation: After reaching the object's position, the robot activates the vacuum pump to create suction and grip the object securely.

d. Release and Reset: After gripping the object successfully, the robot releases it by deactivating the vacuum pump. It then returns to the observation pose to prepare for the next iteration.

Conclusion:

The presented monolithic testing code demonstrates the capabilities of the Niryo robot in detecting, gripping, and releasing red circular objects within a predefined workspace. Through the integration of ROS and the niryo\_robot\_python\_ros\_wrapper, the code enables seamless communication with the robot and facilitates efficient execution of tasks. This script serves as a valuable tool for assessing and validating the robot's core functionalities in real-world applications, such as pick-and-place operations in industrial or research settings.

GitHub Repository: https://github.com/SR42-dev/aravs-niryo-ned2