ROS2 and 3D Vision-Based Automated Pick and Place System for Large-Sized Nuts

Kang-Hee Lee[1]

[1]School of AI & Software, College of AI, Soongsil University, Seoul, South Korea

Abstract.

This paper presents the design and implementation of a Robot Operating System 2 (ROS 2)-based Pick and Place pipeline designed to detect, pick, and transfer industrial large-sized nuts to a destination box. By leveraging RGB-D camera data, point cloud processing, and Eye-to-Hand calibration, the 3D spatial coordinates and grasp poses of target nuts are extracted in real time. MoveIt2 serves as the primary motion planning framework to ensure collision-free trajectory generation, while action-based ROS 2 communication handles task execution and gripper synchronization. The proposed system was validated through simulation and physical hardware testing, demonstrating high grasp reliability and spatial accuracy.

Keywords: ROS2, 3D Vision, Depth-Camera, Pick and Place, Nuts.

[1] *Corresponding author

1. Introduction

1.1 Background

Automating the handling and assembly of heavy metallic parts—such as large nuts and bolts—requires seamless integration between 3D visual perception and robotic manipulation (Billard & Kragic, 2019). Traditional 2D vision systems often struggle under variable lighting conditions and fail to provide the full 6D pose information (position and orientation) necessary for precise spatial manipulation (Du et al., 2021; Duan et al., 2021). Consequently, 3D Spatial Perception using RGB-D depth sensors and point cloud processing has become the standard approach in modern industrial bin-picking and assembly applications (Rusu & Cousins, 2011; Zhuang et al., 2022).

1.2 Motivation and Challenges

Despite recent advancements, picking industrial fasteners introduces unique challenges:

- Metallic surface reflections frequently corrupt depth map estimation in infrared-based sensor arrays (Feng et al., 2020),
- Accurate frame alignment between the vision sensor and the robotic base frame requires rigorous Eye-to-Hand calibration to prevent cumulative execution errors (Shah, 2013; Tsai & Lenz, 1989).
- Deterministic collision-free motion planning must be executed in real time under dynamic industrial workplace constraints (Coleman et al., 2014; Macenski et al., 2022).

1.3 Motivation and Challenges

To address these challenges, this paper presents an end-to-end ROS 2-based Pick and Place pipeline with the following contributions:

- **3D Vision Perception:** real-time 3D position (x, y, z) and pose estimation of metallic nuts using Euclidean clustering and PCA-based orientation extraction from RGB-D point clouds.
- **Coordinate Calibration:** integration of Eye-to-Hand kinematic transformation via `tf2_ros` to maintain dynamic frame synchronization.
- **Trajectory Generation & Execution:** motion planning using MoveIt 2 combined with ROS 2 Action protocols to manage modular end-effector control sequences.

2 System Architecture

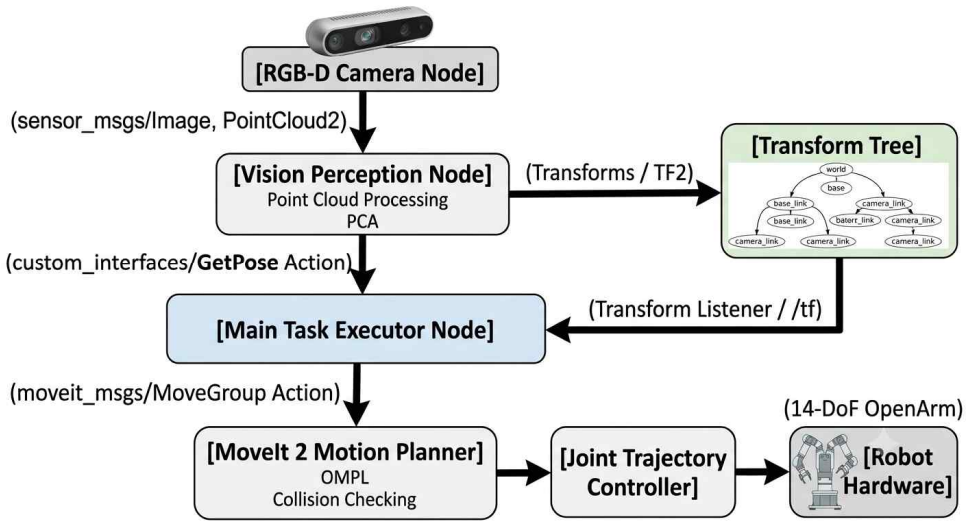
2.1 Hardware Configuration

- **Manipulatr:** 14-DoF OpenArm Robot.
- **End-Effector:** 2-Finger Parallel Industrial Gripper mounted on OpenArm
- **Vision Sensor:** Intel RealSense D455 RGB-D Camera configured in an Eye-to-Hand setup

2.2 ROS2 Node Architecture and Communication

The overall architecture consists of modular ROS2 nodes interacting via Topics, Actions, and Transforms:

Figure 1: Overall architecture of modular ROS2 nodes interacting via topics, actions, and transforms for pick-n-place work



- vision_perception_node: processes point cloud data to filter workspace boundaries, isolate nut geometry, and publish estimated poses.

- task_executor_node: Operates as a Finite State Machine (FSM) overseeing the full pick-and-place lifecycle.

- moveit2_planner: computes joint state trajectories while performing real-time collision avoidance.

3 Vision Perception and Calibaration

3.1 Point Cloud Processing & Pose Estimation

- **PassThrough Filtering:** trims raw point cloud data outside the operational spatial bounds.
- **RANSAC Plane Segmentation:** identifies and subtracts planar table surfaces from the point cloud.
- **Euclidean Cluster Extraction:** groups remaining 3D points to isolate individual nut objects.
- **Pose Estimation:** principal Component Analysis (PCA) determines the center coordinates $P_{cam} = [x_c, y_c, z_c]^T$ and grasp orientation (quaternions).

3.2 Eye-to-Hand Calibration

To transform the nut's position P_{cam} from the camera frame (C) to the robot base frame (B), a homogeneous transformation matrix T_C^B is applied:

$$P_{base} = T_C^B \cdot P_{cam} = \begin{bmatrix} R_C^B & t_C^B \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \quad (1)$$

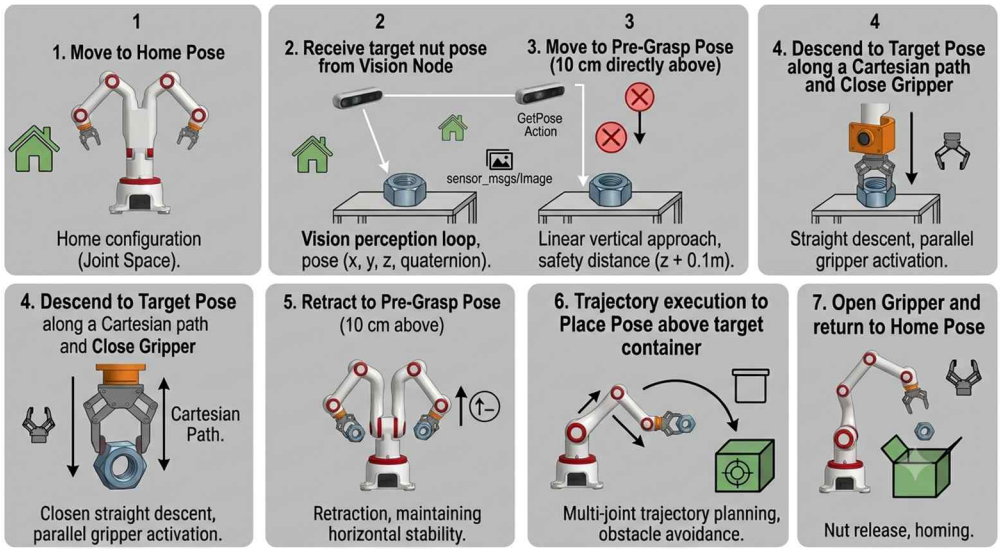
The tf2_ros library dynamically manages frame transformations, keeping the transform tree updated across all active nodes.

4 Pick and Place Control Algorithm

4.1 Grasping Strategy

To avoid unintended collisions, manipulation follows a sequential approach (Approach → Grasp → Retract):

Figure 2: Grasping strategy



4.2 ROS2 Python Core Implementation

The snippet below demonstrates the execution node utilizing MoveItPy and tf2:

Figure 3: ROS2 Python Core code

```python
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import PoseStamped
from moveit_configs_utils import MoveItConfigsBuilder
from moveit.planning_interface import MoveGroupInterface
from tf2_ros import Buffer, TransformListener

class NutPickAndPlaceNode(Node):
    def __init__(self):
        super().__init__('nut_pick_and_place_node')

        # Configure TF2 buffer for frame transformations
        self.tf_buffer = Buffer()
        self.tf_listener = TransformListener(self.tf_buffer, self)

        # Initialize MoveIt 2 interfaces
        self.arm_group = MoveGroupInterface(self, "arm_group")
        self.gripper_group = MoveGroupInterface(self, "gripper_group")

    def execute_pick_and_place(self, target_pose_cam: PoseStamped):
        # 1. Transform Pose from Camera Frame to Robot Base Frame
        try:
            transform = self.tf_buffer.lookup_transform(
                'base_link', target_pose_cam.header.frame_id, rclpy.time.Time())
            target_pose_base = self.tf_buffer.transform(target_pose_cam, 'base_link'
        except Exception as e:
            self.get_logger().error(f"TF Transform Fail: {e}")
            return

        # 2. Calculate Pre-Grasp Pose (10 cm offset in Z axis)
        pre_grasp_pose = PoseStamped()
        pre_grasp_pose.header.frame_id = "base_link"
        pre_grasp_pose.pose = target_pose_base.pose
        pre_grasp_pose.pose.position.z += 0.1  # 10cm offset

        # 3. Move to Pre-Grasp Pose
        self.arm_group.set_pose_target(pre_grasp_pose)
        self.arm_group.plan_and_execute()

        # 4. Cartesian Descent to Target Pose
        self.arm_group.set_pose_target(target_pose_base)
        self.arm_group.plan_and_execute()

        # 5. Close Gripper (Grasp Nut)
        self.gripper_group.set_named_target("close")
        self.gripper_group.plan_and_execute()

        # 6. Retract to Pre-Grasp Pose
        self.arm_group.set_pose_target(pre_grasp_pose)
        self.arm_group.plan_and_execute()

        # 7. Move to Destination Box & Release
```

5 Experimental Results

5.1 Experimental Setup

To validate the proposed ROS 2-based 3D vision perception and trajectory planning framework, experiments were conducted in both simulated and physical environments:

- **Simulation Setup:** a virtual environment was implemented using ROS2 and Gazebo, utilizing Unified Robot Description Format (URDF) models for both the robotic manipulator and the target industrial nut.
- **Physical Setup:** the hardware testbed comprised a 14-DoF OpenArm robotic arm equipped with its parallel gripper and an Eye-to-Hand mounted Intel RealSense D455 RGB-D camera.

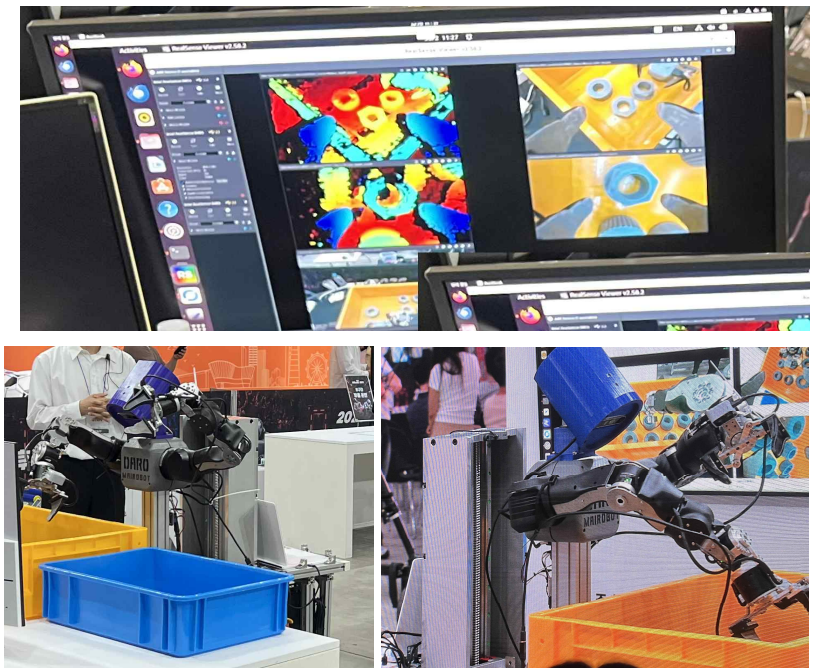
5.2 Performance Analysis

The system was quantitatively evaluated over 30 trial iterations based on three core performance metrics:

- **3D Perception Localization Error:** the 3D vision perception pipeline achieved a mean spatial localization error of 1.8mm, successfully meeting the target accuracy requirement of less than 3.0mm.
- **Pick & Place Success Rate:** the system demonstrated high operational reliability, completing 28 out of 30 trials successfully for a 93.3% success rate, exceeding the 90% target threshold.

- **Average Cycle Time:** the average execution time per complete pick-and-place cycle—from initial target perception to target container placement—was measured at 42.4 seconds, performing well within the 60 second target limit.

Figure 4: Intel RealSense D455 & YOLO7-based Nuts recognition (upper) and OpenArm moving nuts from the yellow box to the blue box (lower).



The system maintained sub-2mm spatial accuracy using the point cloud processing pipeline. The 14-DoF kinematic configuration allowed the OpenArm to execute obstacle avoidance paths while maintaining optimal end-effector orientation during picking. However, specular highlights on shiny metallic nuts occasionally caused temporary depth signal degradation, pointing to the need for deep-learning-based pose estimators in future revisions.

6 Conclusion

This study successfully implemented a ROS 2-based Pick and Place pipeline integrating 3D vision perception and MoveIt 2 trajectory planning tailored for a 14-DoF OpenArm system. The modular ROS 2 action communication layout, dynamic TF frame management, and Cartesian waypoint planning validated the feasibility of using high-DoF open hardware platforms for precision industrial component manipulation.

Acknowledgment

This research was supported by the Ministry of Science and ICT(MSIT), Korea and the Institute of Information & Communications Technology Planning & Evaluation(IITP) under AI University(2026-0-00057, 2026).

References

Billard, A. and Kragic, D. (2019). Trends and challenges in robot manipulation. *Science Robotics*, vol. 4, no. 35, eaat8414.

Coleman, D., Sucan, I., Chitta, S. and Correll, N. (2014). Reducing execution time with the MoveIt! Task Constructor. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Chicago, USA, pp. 1-8.

Du, G., Wang, K. and Lian, S. (2021). Vision-based robotic grasping from 2D images to 3D point clouds: A survey. *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1-16.

Duan, M., Tsai, C. Y. and Chen, C. H. (2021). A review of visual-guided robotic grasping and manipulation. *IEEE Access*, vol. 9, pp. 112345-112364.

Feng, M., Hu, Y. and Zhang, L. (2020). 3D visual perception for metallic reflective objects: A comprehensive review. *Optics and Lasers in Engineering*, vol. 134, 106192.

Macenski, S., Foote, T., Gerkey, B., Lalancette, C. and Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, vol. 7, no. 66, eabm6074.

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, pp. 1-4.

Shah, M. (2013). Solving the AX=YB sensor calibration problem using dual quaternions. *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1240-1245.

Tsai, R. Y. and Lenz, R. K. (1989). A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345-358.

Zhuang, C., Zhang, Y. and Liu, X. (2022). RGB-D sensor-based 3D object pose estimation for industrial robot bin-picking. *Robotics and Computer-Integrated Manufacturing*, vol. 73, 102228.