Sensorized Adaptive Grasping: ROS2 Based Integration of UR3e and Schunk SVH with Force Sensors

Youssef Aboud[1], Andrew Johnson[1], Gidugu Lakshmi Srinivas[1] and Mathias Brandstötter[1]

***Abstract*— Robotic grasping is a critical challenge in automation, requiring precise control to handle objects of varying shapes and fragility. While industrial robotic arms offer reliable motion control, their ability to adapt gripping force dynamically is often limited. This work addresses the need for force-sensitive grasping by integrating the Universal Robot UR3e with the Schunk SVH robotic hand in a ROS2-based framework. The key innovation lies in a real-time force-controlled grasping system, where force sensors embedded in the fingers provide continuous feedback to regulate applied force. The system operates within a closed-loop control structure, ensuring that no additional force is applied to the object once the required force is reached. This prevents deformation or slippage, enabling safer and more adaptive handling. The framework was validated through simulated grasping tasks involving objects such as a ball, a square block, and an apple. Each task tested the system's ability to adjust its grip in response to sensor feedback. The integration process included configuring ROS2-based communication, developing motion planning using MoveIt2, and visualizing robot trajectories in RViz. The UR3e trajectories were tested in Gazebo to simulate grasping interactions before real-world deployment, ensuring reliable execution. Future work will focus on enhancing object detection by integrating computer vision modules into the study. A camera system automatically identifies and localizes objects, reducing reliance on predefined grasping positions. This addition will enable autonomous grasp selection, making robotic manipulation more adaptive in unstructured environments.**

***Index Terms*— ROS2 robotic manipulation, Schunk SVH, Universal Robot UR3e, Adaptive force feedback, closed-loop control system, Sensorised grasping**

I. INTRODUCTION

Robotic grasping remains a fundamental challenge in industrial automation, service robotics, and human-robot interaction. While robotic arms have achieved high precision in motion execution, their ability to handle objects with varying shapes and fragility remains limited. Traditional position-controlled grasping methods lack adaptability, often leading to excessive force application or unstable grip performance [1]. Force-controlled grasping, where tactile sensors provide real-time feedback, enables robots to interact safely and effectively with objects [3]. Several industries, including manufacturing [15], healthcare [17], and logistics [11], demand robotic systems that can grasp objects without predefined parameters. Integrating force sensors in robotic hands enhances adaptability, ensuring secure and precise manipulation without damaging delicate objects [12]. Existing research has explored sensor-driven grasping using various robotic hands [14], but there is still a gap in seamlessly integrating force feedback within ROS2-based control architectures. This work addresses this limitation by developing a real-time force-controlled grasping system for the Universal Robots UR3e and Schunk SVH hand, fully integrated within the ROS2.

Several studies have focused on enhancing robotic grasping through sensor integration and adaptive control. Researchers have explored tactile sensor-based grasping, demonstrating improved grip stability using force sensors on robotic fingers [16], [6]. The Schunk SVH hand has been studied for its human-like dexterity [4], but its potential for adaptive grasping in a ROS2-based environment remains under-explored. While such five-fingered, highly sensitive grippers offer impressive manipulation capabilities, they are not widely adopted in industrial applications due to their complexity and cost. As a result, their use is still largely confined to research environments, where more advanced dexterity and nuanced control are of interest. The Universal Robots UR3e has been widely used in ROS-based applications [13], with works focusing on motion planning using MoveIt2 [7] and real-time execution with RTDE [9]. However, previous studies [18] often rely on position control rather than force feedback, limiting adaptability. Simulated environments like Gazebo have proven effective for testing robotic grasping strategies [8]. Studies integrating ROS and Gazebo have focused on collision-free grasping and trajectory optimization [10], yet a complete pipeline combining force sensing, ROS2, and dynamic control has not been fully realized. Additionally, recent works on sensor fusion for robotic grasping highlight the importance of integrating multiple sensing modalities, including force sensors and vision systems, to achieve optimal grasping strategies [2]. Additionally, learning-based approaches leveraging reinforcement learning have demonstrated improved adaptability by enabling robots to refine their grasping techniques dynamically in unstructured environments [5].

The primary objectives of this paper are to design and implement a ROS2-based control framework that seamlessly integrates the UR3e robotic arm and the SVH five-fingered robotic hand, to develop advanced algorithms for real-time, sensor-driven grip adjustment, and to evaluate the system's performance in dynamic manipulation tasks rigorously. By leveraging the high precision and repeatability of the UR3e, the human-like dexterity and grasping capabilities of the SVH hand, and the adaptability enabled through continuous sensor feedback, this work seeks to push the boundaries of robotic manipulation in complex, unstructured environments.

[1]All authors are with ADMiRE Research Center, Carinthia University of Applied Science, Villach, Austria {youssef.aboud, edujohand001}@edu.fh-kaernten.ac.at and {l.gidugu, m.brandstoetter}@fh-kaernten.at

https://doi.org/10.34749/3061-0710.2025.12

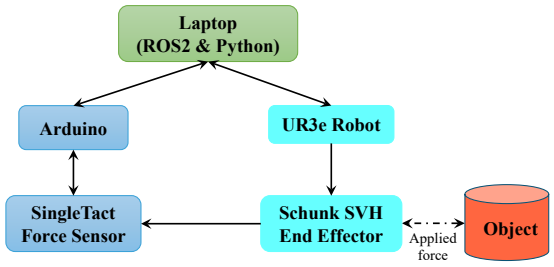

Fig. 1. The hardware and software architecture for ROS2-based adaptive grasping with UR3e, Schunk SVH, Arduino, and force sensors

Through this integration, the paper aims to contribute to more autonomous, flexible, and robust robotic systems capable of performing nuanced tasks in real-world applications.

II. METHODOLOGY

A. Hardware and Software Architecture

The system integrates hardware and software components to enable adaptive force-controlled grasping using the UR3e robotic arm and Schunk SVH end effector. The primary objective is to create a closed-loop control mechanism that dynamically adjusts grip force based on real-time sensor feedback, ensuring reliable and effective manipulation of objects. This architecture is built around the ROS2 middleware, Python for data processing and control, an Arduino microcontroller for sensor communication, and SingleTact force sensors for force measurement, as shown in Fig. 1.

ROS2 is the core communication framework facilitating data exchange between different system components. It provides real-time messaging and distributed computing capabilities that allow nodes running on the control laptop, UR3e robotic arm, and Schunk SVH end effector to interact efficiently. The ROS2 middleware is responsible for processing position commands, reading joint states, and handling interrupts triggered when predefined force thresholds are exceeded. This ensures that grasping actions are dynamically adjusted based on real-time feedback, making it suitable for handling fragile or deformable objects without damage. Python is crucial for system monitoring, sensor data processing, and implementation of control logic. A Python script running on the control laptop continuously listens to incoming data from the force sensors transmitted via the Arduino. The script evaluates whether the grip force exceeds a predefined threshold and, if necessary, publishes an interrupt message to ROS2. This interrupt triggers the system to read the current position of the Schunk SVH fingers and adjust the grip trajectory accordingly. Once the target force level is reached, the Schunk hand is activated to hold the object securely. By leveraging Python's real-time processing capabilities, the system ensures that gripping forces remain within optimal limits, preventing excessive force application and object slippage.

The Arduino microcontroller intermediates the force sensors and the ROS2-based control system. The force sensors are physically attached to the fingers of the Schunk SVH end effector, and their readings are collected and processed by the Arduino. These readings are then forwarded via a serial connection to the control laptop, where Python scripts interpret and analyze the data. The Arduino operates continuously, ensuring that sensor values are relayed in real-time without delay. This low-level sensor acquisition and communication process is essential for the higher-level control algorithms that govern adaptive grasping. A key component in the system's force-sensing capability is the SingleTact force sensor, which provides precise and high-resolution force measurements. The sensor is integrated with a control board that converts analog signals into digital values, ensuring stability and accuracy under different grasping conditions. These digital readings are transmitted to the Arduino through an I2C (Inter-Integrated Circuit) interface, allowing for efficient data acquisition. The control board also handles sensor calibration, ensuring the force readings remain reliable and consistent. Integrating these sensors into the system enables continuous force feedback, allowing for fine-tuned adjustments to the grip strength of the robotic hand.

B. UR3e Integration and Control in ROS2

The UR3e was integrated within the ROS2-based system to provide precise motion control and seamless coordination with the Schunk SVH end-effector. The setup involved configuring network communication, installing the ROS2 driver, validating hardware functionality, and developing custom scripts for motion planning and control. To ensure compatibility with ROS2, the system was set up with Ubuntu and ROS2 Humble. Network communication was established by assigning static IP addresses to both the control PC and the UR3e teach pendant, enabling direct Ethernet-based communication.

The Universal Robots ROS2 driver was obtained from the official GitHub repository[1] and compiled within the ROS2 workspace. A communication program was created on the UR3e teach pendant to enable external command execution. Once the driver was launched on the PC, remote operation of the UR3e was activated, allowing full control via ROS2. To verify hardware functionality, the UR3e was tested using ROS2 service calls and topic-based communication, ensuring proper joint state updates and motion command execution. For simulation, Gazebo was installed, along with the official Universal Robots ROS2-Gazebo integration package[2]. This setup provided a virtual testing environment, allowing trajectory validation before real-world execution. The combination of simulation and physical validation enabled safe experimentation with different control strategies, ensuring reliable robot performance.

C. Motion Control and Trajectory Execution

The motion control of the UR3e robotic arm was implemented using MoveIt2 in ROS2, allowing for joint and

[1] https://github.com/UniversalRobots/Universal_Robots_ROS2_Driver

[2] https://github.com/UniversalRobots/Universal_Robots_ROS2_Gazebo_Simulation

https://doi.org/10.34749/3061-0710.2025.12

Cartesian-based movement execution. Python scripts were developed to control the UR3e's motion using inverse kenimatics and trajectory planning. MoveIt2 provided advanced motion planning features, including collision avoidance and optimized path execution, ensuring smooth and adaptive manipulation. ROS2 nodes were programmed to adjust the robot's movement based on sensor feedback dynamically, enabling precise grasping and interaction with objects. The implementation leveraged action servers to execute motion commands efficiently, ensuring real-time adaptability in robotic grasping tasks.

1) Joint Control Using MoveIt2: Joint-based control regulated individual joint angles, allowing precise control over the UR3e's motion. Instead of defining Cartesian coordinates, this method focused on achieving a specific joint configuration. The MoveIt2 framework computed time-parameterized trajectories that guide each joint to its target position while respecting joint-level velocity and acceleration constraints. Inverse kinematics was performed using the default KDL solver in MoveIt2, which provides joint configurations that are locally optimal in terms of minimal displacement from the current joint state. This ensures smooth, continuous motion that is well-suited for real-time grasping tasks. The ROS2 action server sent motion commands, ensuring reliable execution. This approach is particularly useful for structured tasks such as pick-and-place operations, where predefined joint configurations are essential. The combined joint and cartesian control implementation process is provided as a pseudo-code, as shown in Algorithm 1.

Algorithm 1 Combined Pseudo Code for UR3e Joint & Cartesian Control Using MoveIt2 in ROS2

1: **Initialize** the ROS2 system and create a node for MoveIt-based control.
2: **Establish** an Action Client for MoveIt2's `MoveGroup` action server.
3: **Wait** for the `/move_action` server to become available.
4: **Define a function for Joint Control (MoveJ):**
- Set target joint positions.
- Specify velocity and acceleration scaling factors.
- Create MoveIt2 joint constraints and assign them to UR3e joints.
- Send the `MoveJ` command via MoveIt2 Action Client.
- Execute `MoveJ` with chosen velocity and acceleration.
5: **Define a function for Cartesian Control (MoveL):**
- Set a target position in Cartesian space (x, y, z).
- Apply end-effector constraints for straight-line motion.
- Use a bounding box or waypoints for accurate positioning.
- Send the `MoveL` command to the action server.
- Execute `MoveL` with controlled speed and accuracy.
6: **Keep** the ROS2 node running for continuous operation.
7: **Shutdown** the node upon completion.

2) Cartesian Control Using MoveIt2: Cartesian control was implemented to enable the UR3e's end-effector to reach specific positions in Cartesian space (X, Y, Z) rather than following predefined joint angles. This approach relied on the inverse kenimatics to calculate the required joint positions dynamically. MoveIt2 generated collision-free trajectories, ensuring smooth and precise motion. Cartesian constraints, such as bounding boxes and position constraints on the wrist, were applied to maintain accuracy. This method is particularly beneficial for applications with critical end-effector positioning, such as assembly tasks and object manipulation.

Fig. 2 demonstrates motion control of a UR3e robot using ROS2 and MoveIt2, showcasing both joint-based and Cartesian-based control methods. In the left section (a), MoveJ is used for joint-space motion planning, where the robot moves through a series of predefined joint angles for precise articulation. In the right section (b), MoveL is applied for Cartesian-space motion, ensuring the end-effector follows a straight-line trajectory in 3D space. The terminal outputs confirm the successful execution of both control commands, with MoveJ handling overall joint positioning and MoveL ensuring smooth linear movements. These approaches allow flexible motion planning, depending on the task requirements, such as obstacle avoidance or precise end-effector placement.

D. Schunk SVH end-effector

Integrating the Schunk SVH five-fingered hand within the ROS2-based system followed a structured approach to ensure reliable operation and precise control. The integration process consisted of system setup, library installation, hardware validation, and the development of custom scripts for joint control. The software environment was configured by installing Ubuntu with ROS2 Humble, ensuring compatibility with the Schunk SVH ROS2 driver. Dependencies were verified and installed to facilitate seamless communication between ROS2 and the end effector. The Schunk SVH ROS2 driver was obtained from the official GitHub repository[3]. Due to limited documentation of the SVH ROS2 driver, script

[3]`https://github.com/SCHUNK-SE-Co-KG/schunk_svh_ros_driver`

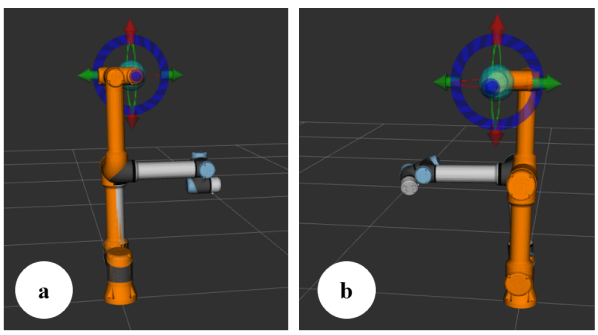

Fig. 2. Comparison of joint-space (MoveJ) and Cartesian-space (MoveL) motion control for a UR3e robot using ROS2 and MoveIt2

https://doi.org/10.34749/3061-0710.2025.12

development required debugging and adaptation of provided examples. These custom scripts formed the foundation for reliable communication between ROS2 and the SVH hand. The hand was connected via USB, and communication with the ROS2 framework was achieved without issues. The provided ROS2 script examples were adapted to control each joint of the Schunk SVH hand. These scripts were used to execute test sequences, verifying the accuracy and repeatability of joint movements. The inverse kenimatics was applied to the end effector to accurately position the end effector (EE) for the pick-and-place tasks. The transformation matrix represents the position and orientation of the end effector relative to the target object, which was used as input to the inverse kenimatics algorithm. The solution provides the necessary position and orientation of the end effector to ensure that the robot's arm places it precisely at the required location. This calculation considers the robot's physical constraints and ensures that the end effector reaches the target with the correct pose without recomputing the inverse kinematics of the entire arm.

E. Force sensors

1) Sensor Selection and Setup: A SingleTact capacitive force sensor was selected due to its high sensitivity and compact form factor, making it suitable for integration into the Schunk SVH robotic hand. The sensor is small enough to be affixed to the inner gripping surfaces of the fingers, enabling direct measurement of contact forces during object manipulation. The sensor was connected to its control board, which provided signal conditioning and a digital output accessible via an I2C interface. An Arduino Uno was used to interface between the control board and ROS2. To ensure reliable force measurements, the sensor was placed on the distal phalanx of the robotic thumb, as shown in Fig. 3. This location was chosen to ensure contact with the sensor while grasping objects of varying shapes.

2) Serial Communication and Data Parsing: The force sensor data was acquired using an Arduino Uno microcontroller, which then transmits the sensor readings at a 57600 baud rate over a serial connection. The flowchart provides the communication process, as shown in Fig. 4. The Arduino firmware transmits raw integer values, which are then parsed and processed in a Python-based ROS2 node (stopper.py) running on an Ubuntu-based control system. The node reads

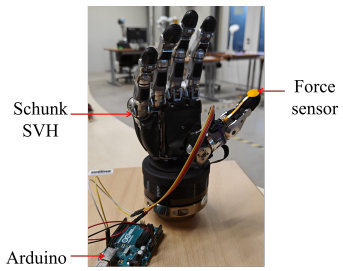

Fig. 3. The placement of SingleTact force sensor on the distal phalanx of SVH

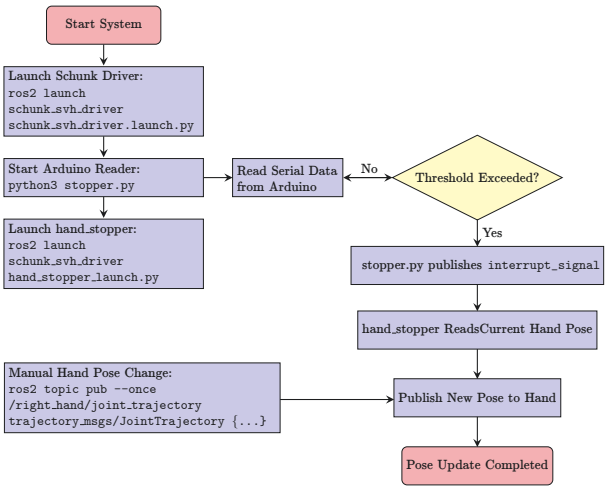

Fig. 4. The flowchart of Schunk SVH communication using ROS2

sensor data in real time and triggers a hand-stop interrupt signal once the force reaches a predetermined threshold. A rising edge detection mechanism ensures that an interrupt signal is only published when the sensor value exceeds a threshold for the first time, preventing redundant commands.

3) ROS2 Node hand_stopper: Once an interrupt signal is triggered by stopper.py, the hand_stopper node executes a defined script to halt the hand's movement immediately. This is achieved by retrieving the most recent joint positions from the /joint_states topic and sending this as a new trajectory command to maintain the current pose.

The node subscribes to /joint_states to continuously update an internal dictionary containing the latest joint positions of the right hand. When an interrupt is received, the node:

- Checks if valid joint states have been recorded. If no valid positions are available, it does not issue a stop command to avoid unintended behavior.
- Retrieves the most recent joint positions for the right-hand fingers.
- Constructs a JointTrajectory ROS2 message with these positions as the target.
- Publishes this trajectory to /right_hand/joint_trajectory, ensuring the hand holds its last known position.

To achieve a smooth stop, the trajectory message includes a short time delay (e.g., 50ms) in order to prevent high jerk. This ensures a rapid but controlled halt, preventing excessive force while maintaining stability. The hand remains in this position until a new command is issued, preventing unnecessary fluctuations in grip force.

4) Performance Analysis: The system mitigated excessive gripping force by dynamically adjusting the hand's pose in response to high-pressure readings. Key results include a significant reduction in grasping force, ensuring the safe handling of fragile objects. Additionally, real-time data processing enabled immediate response to pressure fluctuations, allowing for precise and timely grip adjustments. Adaptive pose control enhanced the gripper's efficiency and simplified

https://doi.org/10.34749/3061-0710.2025.12

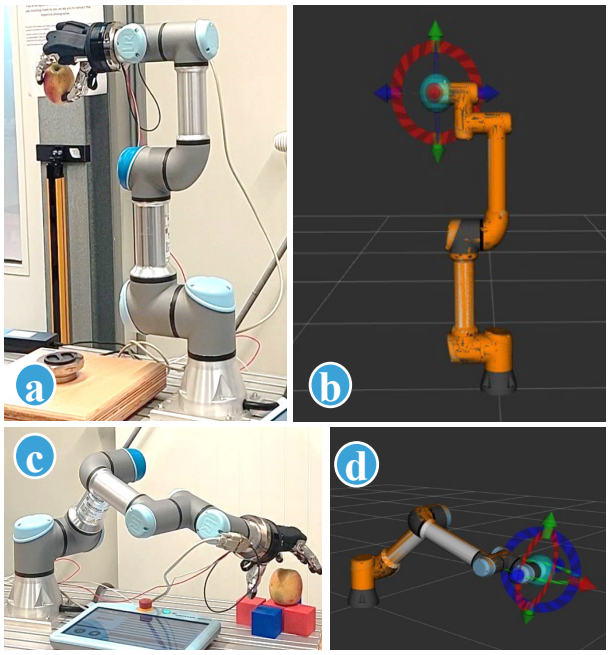

Fig. 5. UR3e home and object-picking position in real and simulated environments

overall control, making it more responsive to varying object shapes and material properties.

III. RESULTS AND DISCUSSIONS

1) UR3e Motion Execution in Simulation and Real-World: The validation of robotic grasping systems relies on ensuring that simulated motion planning closely mirrors real-world execution. To achieve this, the UR3e robotic arm was first tested in a simulated environment before deploying the same motion sequences in real-world trials. The Gazebo simulation platform generated and refined motion trajectories, ensuring that the robotic arm's planned movements were accurate and feasible. Two positions are provided to demonstrate the alignment between simulation and reality. Fig. 5 (a) captures the UR3e in a real-world setup at its home position, while Fig. 5 (b) presents the corresponding simulated model in Gazebo. Similarly, Fig. 5 (c) and (d) showcase the robot at the object-picking position in both real and simulated environments. The consistency in joint configurations and movement sequences across both domains highlights the effectiveness of the ROS2-based motion control system in ensuring reliable robotic manipulation. During execution, the UR3e robot follows a structured sequence, beginning from a predefined home position before transitioning into object interaction tasks. The home position establishes a stable and repeatable starting point, improving trial consistency. From this state, the robot moves towards the target object following a planned trajectory, ensuring smooth transitions and avoiding unintended deviations. The inverse kenimatics solver calculates the optimal joint configurations, which are first validated in Gazebo before real-world execution. This step ensures that the simulated robot's movement precisely mirrors the physical robot's behavior, reducing potential errors during deployment. The comparison between simulation and real-world execution confirms the robustness of trajectory planning and motion replication. The UR3e successfully follows pretested motion paths, demonstrating the reliability of ROS2-based control for adaptive robotic applications. The seamless transition from simulation to real execution minimizes risk, improves efficiency, and ensures safe and repeatable grasping operations.

2) Force-Controlled Grasping and Object Handling: As the Schunk SVH hand approaches the target, it gradually applies force until reaching a predefined threshold, ensuring a controlled grasp. The threshold varies based on the object's properties, with 5 N used for this demonstration, as shown in Fig. 6. Excessive force application stops once the threshold is met, and the robot moves toward the endpoint while maintaining a stable grip. Upon reaching the target, the robotic hand gradually releases the force, ensuring smooth object placement. This adaptive control prevents slippage, reduces the risk of damage, and ensures secure handling. The results confirm the effectiveness of the force-controlled grasping strategy, where the robotic hand dynamically adjusts its grip to accommodate different objects. The system prevents excessive force while maintaining stability, demonstrating the ROS2-based closed-loop control's reliability. The force trajectory in Fig. 6 highlights stable gripping and controlled release, validating its suitability for adaptive robotic manipulation.

3) Sequential Adaptive Grasping Demonstration: The sequence illustrates the adaptive grasping process of a robotic hand, showcasing its transition from an open resting state to precise object manipulation, as shown in Fig. 7. The robotic hand is initially fully open, relaxed, and ready for action. It then spreads its fingers to maximum extension, demonstrating flexibility before gradually moving towards a half-closed state, signaling the beginning of a grasping motion. As the thumb flexes inward, the hand adjusts its posture for an impending grasp. During this transition, the robotic hand momentarily forms expressive gestures, including the "rock and roll" and "peace sign," highlighting its dexterity and human-like articulation. Moving beyond expressive gestures, the hand focuses on functional grasping, positioning itself precisely over an object in the hovering phase, preparing for contact. It then executes a precision grip, delicately

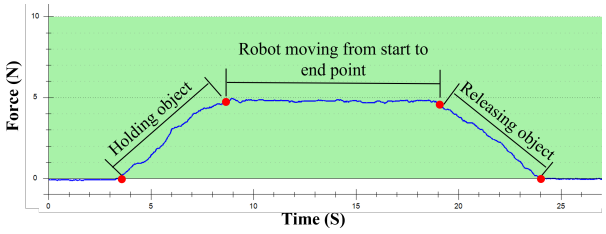

Fig. 6. Force profile of adaptive grasping and object handling

https://doi.org/10.34749/3061-0710.2025.12

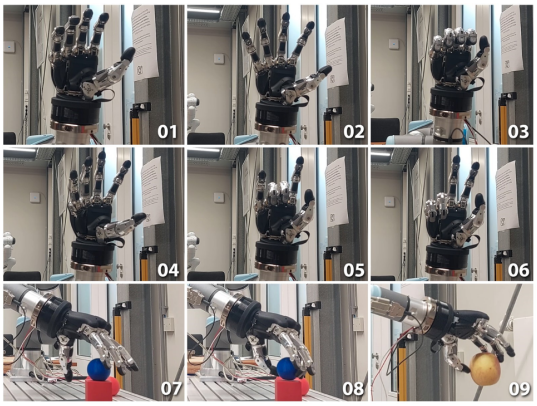

Fig. 7. Sequential demonstration of robotic hand gestures and grasping

securing a small spherical object, emphasizing controlled finger movements. Finally, the robotic hand can gently handle fragile objects, carefully lifting an apple, ensuring a secure yet sensitive grasp. This sequence effectively conveys the robot's capability for expressive gestures and intricate object manipulation, reinforcing its potential for advanced robotic applications.

IV. Conclusion

This work demonstrated a ROS2-based force-controlled grasping system, integrating the UR3e robotic arm and Schunk SVH hand with force sensors for adaptive and precise object manipulation. A closed-loop control mechanism was implemented, where force feedback from the sensors dynamically regulated grip strength, ensuring secure yet non-damaging handling of objects. The system was fully developed within ROS2, utilizing MoveIt2 for motion planning, RTDE for real-time execution, and Gazebo simulations for safe validation before deployment. The experimental validation demonstrated that the robotic hand successfully adjusted its grip in response to sensor feedback, preventing excessive force application while maintaining a stable grasp. Simulated trajectories in Gazebo closely mimicked real-world execution, confirming the accuracy and reliability of the ROS2-based motion planning and control framework. The results highlight the effectiveness of the force-regulated grasping strategy, allowing the system to handle fragile and rigid objects with appropriate force levels. The force profile analysis showed smooth transitions in gripping, transporting, and releasing objects, validating the system's adaptability. The structured motion execution, starting from a home position to object interaction, further ensured repeatability and consistency across trials. Transferring simulation-based planning to real-world execution minimized errors and enhanced efficiency, making the approach viable for various robotic manipulation tasks.

Future work will integrate computer vision-based object recognition to enable autonomous pick-and-place operations. This will allow the robot to adjust force thresholds based on detected object properties dynamically, improving adaptability. Expanding the system's multi-finger coordination will enhance grasping dexterity, making it suitable for complex industrial automation, assistive robotics, and logistics applications. The proposed approach provides a scalable and efficient solution for adaptive robotic grasping in unstructured environments.

References

[1] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 348–353.

[2] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on robotics*, vol. 30, no. 2, pp. 289–309, 2013.

[3] M. R. Cutkosky *et al.*, "On grasp choice, grasp models, and the design of hands for manufacturing tasks." *IEEE Transactions on robotics and automation*, vol. 5, no. 3, pp. 269–279, 1989.

[4] C. Della Santina, C. Piazza, G. Grioli, M. G. Catalano, and A. Bicchi, "Toward dexterous manipulation with augmented adaptive synergies: The pisa/iit softhand 2," *IEEE Transactions on Robotics*, vol. 34, no. 5, pp. 1141–1156, 2018.

[5] B. K. Farkas, P. Galambos, and K. Széll, "Advances in autonomous robotic grasping: An overview of reinforcement learning approaches," in *2024 IEEE 6th International Symposium on Logistics and Industrial Informatics (LINDI)*. IEEE, 2024, pp. 000 213–000 220.

[6] Z. Kappassov, J.-A. Corrales, and V. Perdereau, "Tactile sensing in dexterous robot hands," *Robotics and Autonomous Systems*, vol. 74, pp. 195–220, 2015.

[7] Z. Kingston and L. E. Kavraki, "Robowflex: Robot motion planning with moveit made easy," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3108–3114.

[8] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. Ieee, 2004, pp. 2149–2154.

[9] A. P. Lindvig, I. Iturrate, U. Kindler, and C. Sloth, "ur_rtde: An interface for controlling universal robots (ur) using the real-time data exchange (rtde)," in *2025 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2025, pp. 1118–1123.

[10] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 734–743.

[11] M. Q. Mohammed, L. C. Kwek, S. C. Chua, A. Al-Dhaqm, S. Nahavandi, T. A. E. Eisa, M. F. Miskon, M. N. Al-Mhiqani, A. Ali, M. Abaker, *et al.*, "Review of learning-based robotic manipulation in cluttered environments," *Sensors*, vol. 22, no. 20, p. 7938, 2022.

[12] U. E. Ogenyi, J. Liu, C. Yang, Z. Ju, and H. Liu, "Physical human–robot collaboration: Robotic systems, learning methods, collaborative strategies, sensors, and actuators," *IEEE transactions on cybernetics*, vol. 51, no. 4, pp. 1888–1901, 2019.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.

[14] S. W. Ruehl, C. Parlitz, G. Heppner, A. Hermann, A. Roennau, and R. Dillmann, "Experimental evaluation of the schunk 5-finger gripping hand for grasping tasks," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*. IEEE, 2014, pp. 2465–2470.

[15] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*. Springer, 2008, vol. 200.

[16] J. Tegin and J. Wikander, "Tactile sensing in intelligent robotic manipulation–a review," *Industrial Robot: An International Journal*, vol. 32, no. 1, pp. 64–70, 2005.

[17] W. Wang, J. Wang, Y. Luo, X. Wang, and H. Song, "A survey on force sensing techniques in robot-assisted minimally invasive surgery," *IEEE Transactions on Haptics*, vol. 16, no. 4, pp. 702–718, 2023.

[18] B. Zhang, J. Zhou, Y. Meng, N. Zhang, B. Gu, Z. Yan, and S. I. Idris, "Comparative study of mechanical damage caused by a two-finger tomato gripper with different robotic grasping patterns for harvesting robots," *Biosystems Engineering*, vol. 171, pp. 245–257, 2018.

https://doi.org/10.34749/3061-0710.2025.12