Assistive Feeding Robot

Adithya Rajendran
Ms Robotics
Northeastern University
Boston, USA
rajendran.ad@northeastern.edu

Kevin Jason
Ms Robotics
Northeastern University
Boston, USA
jason.ke@northeastern.edu

Lokesh Chandrasekar
Ms Robotics
Northeastern University
Boston, USA
chandrasekar.l@northeastern.edu

Ahilesh Rajaram
Ms Robotics
Northeastern University
Boston, USA
rajaram.a@northeastern.edu

Pradeep Sivaa Aiyanar
Ms Robotics
Northeastern University
Boston, USA
aiyanar.p@northeastern.edu

***Abstract* —** This paper presents a simulation-based assistive feeding robot system designed for individuals with upper-limb impairments. The system employs a Universal Robots UR5e manipulator to execute spoon-to-mouth feeding trajectories while maintaining safety through continuous impedance-based monitoring. A hybrid motion-planning strategy is introduced that combines joint-space point-to-point interpolation for large reconfigurations with Cartesian linear interpolation for precise feeding motions, addressing the fundamental trade-off between singularity avoidance and end-effector orientation preservation. The impedance monitoring framework estimates contact forces from position tracking errors without requiring external force sensors, enabling emergency stops when estimated forces exceed 25 N. Experimental validation in Gazebo simulation demonstrates successful trajectory execution with peak interaction forces of 19.12 N and zero safety violations under nominal operating conditions.

I. INTRODUCTION

1. *Problem Context*

Approximately 6.8 million Americans live with upper-limb impairments that compromise their ability to perform essential activities of daily living, including self-feeding [1]. For these individuals, every meal requires caregiver assistance, creating significant dependencies that affect both quality of life and caregiver burden. Robotic feeding assistance offers a promising pathway toward restoring independence, yet the intimate nature of feeding where a robot-held utensil operates centimeters from a user's face demands exceptional safety guarantees.

2. *Limitations of Existing Approaches*

The Obi robotic arm and Meal Buddy system are examples of commercial feeding devices that rely on pre-programmed, open-loop trajectories. Without detecting or reacting to the user's condition, these systems carry out predetermined motion sequences. There are two significant drawbacks to this strategy. First, the robot continues on its intended course even if the user moves unexpectedly while feeding—for example, by coughing, turning their head, or making an involuntary motion—which could result in an accident or injury. Second, these systems are unable to modulate their behavior to ensure gentle interaction because they lack force feedback, which makes it impossible for them to detect when contact occurs.

3. *Research Contributions*

This work addresses these limitations through two primary contributions. First, we develop a hybrid motion planning strategy that intelligently switches between joint-space and Cartesian-space trajectory generation based on task phase requirements. This approach resolves the inherent tension between singularity avoidance during large robot reconfigurations and orientation preservation during critical feeding motions. Second, we implement an impedance-based safety monitoring framework that estimates contact forces from position tracking errors, enabling collision detection and emergency stopping without requiring expensive force/torque sensors. Together, these contributions enable smooth, safe feeding trajectories validated through comprehensive simulation experiments.

II. METHOD AND APPROACH

1. *System Architecture Overview*

The complete system architecture integrates motion planning, trajectory execution, and safety monitoring within a ROS 2 framework. Figure 1 illustrates the control pipeline, which processes task-space waypoint specifications through inverse kinematics, trajectory generation, and real-time execution while continuously monitoring for unsafe conditions.

The software stack comprises ROS 2 Jazzy as the middleware layer, Gazebo Harmonic for physics simulation, MoveIt 2 for motion planning and inverse kinematics computation, and ros2_control for joint-level trajectory execution. The simulation environment includes a UR5e manipulator, a table with food plate, and a static human model positioned at a realistic feeding distance of 0.72 meters from the robot base.

Fig 1: Simulation Environment

2. *Control Pipeline Architecture*

The control pipeline operates as follows. Waypoints are specified in Cartesian space as end-effector poses comprising position (x, y, z) and orientation (roll, pitch, yaw). These specifications are passed to the MoveIt 2 inverse kinematics service, which computes corresponding joint configurations. The hybrid motion planner then generates time-parameterized joint trajectories, which are published to the ros2_control scaled joint trajectory controller. Gazebo simulates the robot dynamics and provides joint state feedback. Concurrently, the impedance monitoring module continuously estimates contact forces and triggers emergency stops if safety thresholds are exceeded.

Data flows through ROS 2 topics and services. The /joint_states topic provides real-time joint positions, velocities, and efforts at approximately 100 Hz. The /scaled_joint_trajectory_controller/joint_trajectory topic receives commanded trajectories. Transform (TF) lookups provide end-effector pose estimates for the impedance calculator.

3. *Inverse Kinematics with Semantic Seeding*

The UR5e is a 6-degree-of-freedom manipulator, meaning that for most reachable end-effector poses, multiple joint configurations (IK solutions) exist. The choice among these solutions significantly impacts motion quality and safety. Our system interfaces with MoveIt 2's /compute_ik service, which accepts a target pose and a seed joint configuration that biases the numerical solver toward a particular solution region.

We employ a semantic seeding strategy where the seed configuration is selected based on the current task phase. For waypoints involving reaching toward the food plate (located to the robot's left), we provide a seed state with positive shoulder rotation and extended elbow, biasing the solver toward "elbow-out" configurations that naturally reach leftward. For waypoints approaching the user's mouth (located forward), we provide a centered seed configuration that orients the arm directly forward. This phase-aware seeding produces consistent, predictable joint trajectories and prevents the solver from selecting configurations that would require unnecessary joint rotations between consecutive waypoints.

The IK request includes a 2-second timeout and uses the current joint state as the default seed when no phase-specific seed is appropriate. If the solver fails to find a solution, the system logs the failure and skips the waypoint rather than executing a potentially unsafe motion.

2.4 Hybrid Motion Planning Strategy

A fundamental challenge in feeding robot trajectory generation is the trade-off between singularity avoidance and end-effector orientation preservation. Joint-space interpolation generates smooth trajectories that avoid kinematic singularities but does not guarantee straight-line end-effector motion, potentially tilting a utensil and spilling its contents. Cartesian-space interpolation maintains straight-line paths and constant orientation but can fail or produce erratic behavior when the trajectory passes near singular configurations.

Our hybrid approach addresses this trade-off by selecting the interpolation method based on task phase requirements. The feeding sequence is divided into transport phases and feeding phases, each with distinct kinematic characteristics.

2.4.1 Transport Phase (Joint-Space Planning)

During transport phases specifically, motions from the home position to a safe intermediate height and the return motion at the end of feeding the system employs joint-space point-to-point (PTP) interpolation. The trajectory is generated by computing the joint configuration at the target waypoint via inverse kinematics and interpolating linearly in joint space from the current configuration to the target. This approach is robust to singularities because it never requires computing the Jacobian inverse or commanding Cartesian velocities. The resulting end-effector path is curved in Cartesian space, but this is acceptable during transport when the utensil is either empty or safely distant from the user.

2.4.2 Feeding Phase (Cartesian-Space Planning)

During feeding phases motions between the plate and the user's mouth the system employs Cartesian linear interpolation. The trajectory is generated by computing intermediate end-effector poses along a straight line from the current pose to the target, then solving inverse kinematics for each intermediate pose. This ensures the end-effector follows a straight path with constant orientation, keeping the utensil level to prevent spilling. The intermediate poses are spaced at

1-centimeter intervals (the max_step parameter) to ensure smooth motion.

The switching logic is implemented as follows:

if waypoint_name in ["home", "safe_height", "return_center"]:

trajectory = generate_joint_space_trajectory(target_pose, duration)

else:

trajectory = generate_cartesian_trajectory(target_pose, duration)

This hybrid strategy provides the best of both approaches: robust, singularity-free motion during large reconfigurations and precise, orientation-preserving motion during the critical feeding-segment.

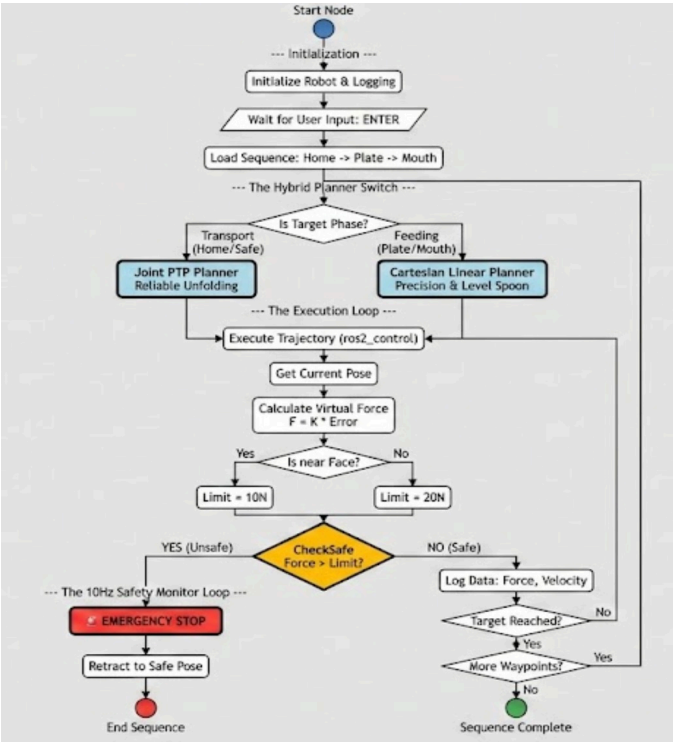

Fig 2: Hybrid-Motion Planning Architecture

2.5 Impedance-Based Safety Monitoring

The safety monitoring framework is designed to detect collisions and unsafe contact forces without requiring external force/torque sensors. This sensorless approach reduces hardware costs and complexity while providing real-time safety enforcement.

2.5.1 Theoretical Foundation

The framework is based on the principle of virtual model control, which models the robot's interaction with its environment as a mass-spring-damper system. In this model, if the robot is commanded to a target position but encounters an obstacle preventing it from reaching that position, the resulting position error can be interpreted as the compression of a virtual spring. The force that would result from this compression provides an estimate of the actual contact force.

The fundamental equation governing this estimation is:

$F_{virtual} = K \cdot (x_{target} - x_{actual})$

$F_{virtual}$ = Estimated 3D force vector K = Diagonal stiffness matrix x_{target} = Commanded end-effector position x_{actual} = Current end-effector position (from TF transforms) Stiffness Matrix: $K = diag(100, 100, 100)$ N/m Interpretation: Each meter of position error in any Cartesian direction corresponds to 100 Newtons of estimated force. Rationale: This value was selected based on typical impedance control parameters for collaborative robots in human-interaction scenarios [2].

2.5.2 Adaptive Stiffness Gains

A fixed stiffness value is insufficient for feeding tasks because different task phases have different safety requirements. During transport motions far from the user, larger position errors are acceptable and should not trigger false alarms. During feeding motions near the user's face, even small deviations may indicate contact and warrant immediate response. We address this through phase-dependent stiffness scaling. The base stiffness matrix is multiplied by a gain factor that varies with task phase:

Task Phase	Gain Factor	Effective Stiffness	Rationale
Home/Transport	0.3	30 N/m	Large motions, high tolerance
Approach	0.8	80 N/m	Standard tracking
Contact (near face)	0.5	50 N/m	Maximum sensitivity
Retract	0.6	60 N/m	Careful withdrawal

Table 1: Phase-Dependent Stiffness Scaling Parameters

The "Contact" phase uses a moderate gain (0.5) rather than the highest gain because we want the system to be most sensitive to small errors near the user's face. A lower stiffness means smaller position errors produce significant force estimates, triggering the safety stop earlier.

2.5.3 Collision Detection and Emergency Stop

The safety monitoring loop executes at 10 Hz, synchronized with the trajectory execution loop. At each iteration, the system performs the following steps:

I. Position Query

The current end-effector position is obtained via TF lookup from the base_link frame to the tool0 frame.

II. Error Computation

The position error vector is computed as the difference between the current target waypoint and the actual position.

III. Force Estimation

The virtual force is computed by multiplying the position error by the phase-scaled stiffness matrix.

IV. Signal Filtering

A rolling mean filter with a window size of 8 samples (0.8 seconds) is applied to the force magnitude. This filtering is essential because the Gazebo physics engine introduces high-frequency numerical noise that can cause spurious force spikes. The filter smooths these artifacts while preserving genuine force trends.

V. Threshold Comparison

If the filtered force magnitude exceeds 25 Newtons, a safety violation is detected.

VI. Emergency Stop Execution

Upon detecting a safety violation, the system immediately publishes a "hold position" trajectory is a trajectory with a single point at the current joint configuration with zero velocity. This causes the robot to decelerate and hold its current position rather than continuing toward the target.

The 25-Newton threshold is derived from ISO/TS 15066 guidelines for collaborative robot safety [3], which specify maximum permissible forces for transient contact with various body regions. For the face and head region, transient contact forces should not exceed approximately 65 N for impact or 25 N for quasi-static loading. We use the more conservative quasi-static limit because our feeding motions are slow and deliberate.

The emergency stop is implemented in the MoveItIKController class:

```
def stop(self):
    """Execute emergency stop by holding current position."""
        if self.current_joint_state is None:
        return
        stop_trajectory = JointTrajectory()
    stop_trajectory.joint_names = self.joint_names
        point = JointTrajectoryPoint()
Point.positions = self.current_joint_state.positions.tolist()
        point.velocities = [0.0] * 6
    point.time_from_start.sec = 0
    point.time_from_start.nanosec = 500000000  # 0.5 seconds
    stop_trajectory.points.append(point)
    self.trajectory_pub.publish(stop_trajectory)
```

2.5.4 Limitations of Sensorless Force Estimation

It is important to acknowledge the limitations of this approach. The virtual force estimate is based on position error, which can arise from sources other than contact—for example, trajectory tracking lag, actuator saturation, or modeling errors. The system cannot distinguish between these causes, meaning it may produce false positives (stopping when no collision occurred) or false negatives (failing to detect soft contact that does not significantly impede motion). For deployment on physical hardware, integration of a force/torque sensor would provide more reliable collision detection. However, for simulation validation and as a first-layer safety mechanism, the sensorless approach provides meaningful protection.

2.6 Feeding Sequence Definition

The complete feeding sequence comprises eight waypoints traversed in order:

I. Home : Initial resting configuration with arm folded.

II. Safe Height : Elevated intermediate position for collision-free transport.

III. At Plate : Position above the food plate, ready for scooping.

IV. Pre-Person : Intermediate position during approach to the user.

V. Near Person : Closer approach to user's head region.

VI. At Mouth : Final delivery position near the user's mouth.

VII. Retract from Face : Withdrawal position after feeding.

VIII. Return to Home : Final resting configuration.

Each waypoint is specified in the robot configuration file with position (x, y, z in meters) and orientation (roll, pitch, yaw in radians). The orientations are configured to keep the end-effector (and attached utensil) level throughout the feeding segment, preventing spillage.

2.7 Collision Environment Setup

The MoveIt 2 planning scene is populated with collision objects representing the feeding environment. The human model comprises a box primitive for the torso (dimensions 0.16 × 0.30 × 0.38 m), a sphere for the head (radius 0.15 m), and cylinder primitives for the arms. Furniture includes the table surface, chair seat, and chair backrest. A cylinder primitive represents the food plate.

These collision objects serve two purposes. First, they enable MoveIt's collision checking during Cartesian path planning, preventing the generation of trajectories that would cause robot-environment collisions. Second, they define the human safety zone a spherical region of radius 0.22 m centered on the head position within which the adaptive stiffness gains are reduced to increase safety sensitivity.

III. RESULTS

3.1 Experimental Setup

We conducted feeding sequence experiments in Gazebo Harmonic simulation with the UR5e robot model as mentioned in the video [6]. Each experiment executed the complete 8-waypoint feeding sequence, with trajectory execution times of 8-10 seconds per segment. The simulation ran at real-time factor 1.0, ensuring that temporal dynamics accurately reflected physical behavior. Data logging captured joint states (positions, velocities, efforts) and virtual force estimates at 10 Hz throughout each trial.

3.2 Nominal Operation Performance

Figure 2 presents the performance analysis dashboard for a representative successful trial. The four-panel visualization shows joint positions, joint velocities, estimated interaction force, and motion smoothness (jerk).

3.2.1 Joint Position Trajectories

The top-left panel shows the six joint angles over the 82.51-second execution. All joints exhibit smooth, continuous trajectories with no discontinuities or sudden jumps. The trajectories reflect the sequential waypoint execution, with distinct motion segments corresponding to each phase of the feeding sequence. Joints 1 and 2 (shoulder pan and shoulder lift) show the largest excursions as they are primarily responsible for positioning the arm in the workspace.

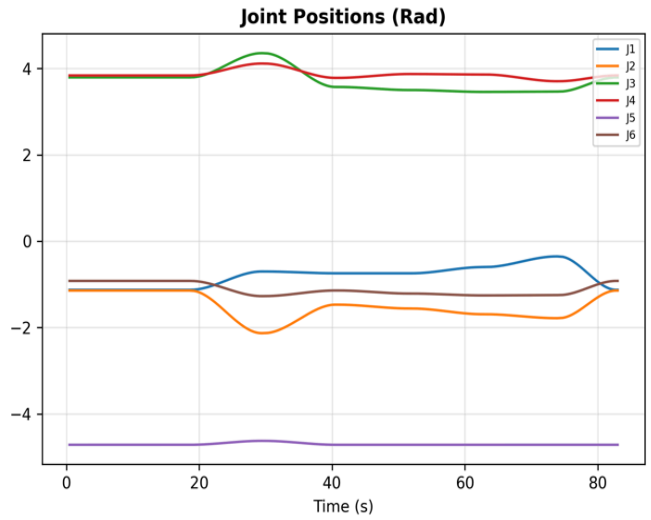

Fig 3: Plot for Joint Positions

3.2.2 Joint Velocity Profiles

The top-right panel shows joint velocities, which exhibit smooth sinusoidal-like profiles characteristic of properly time-parameterized trajectories. The peak velocity of 0.148 rad/s is well below the UR5e's maximum joint velocity limits (±3.14 rad/s for shoulder joints, ±6.28 rad/s for wrist joints), indicating conservative, safe motion speeds. The absence of velocity spikes confirms that trajectory interpolation is functioning correctly.

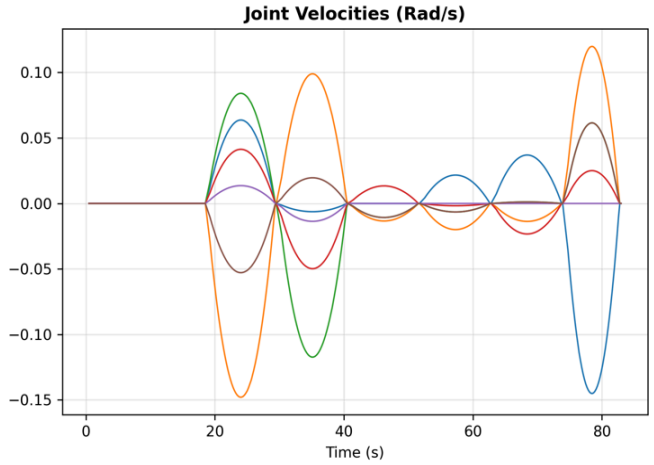

Fig 4: Plot for Joint Velocities

3.2.3 Interaction Force Estimation

The bottom-left panel shows the critical safety metric the estimated virtual force magnitude over time. The horizontal dashed line indicates the 25 N safety threshold. The force profile reveals distinct phases of the feeding sequence:

I. During the initial transport phase (0-20 s), force remains near zero as the robot moves freely through space.

II. During plate approach (20-35 s), force increases to approximately 19 N as the robot approaches the table region, indicating increased tracking effort.

III. During the feeding phase (35-70 s), force oscillates between 1-12 N as the robot executes precise motions near the plate and user.

IV. During return transport (70-82 s), force increases again to approximately 15 N.

Critically, the peak force of 19.12 N remains 24% below the 25 N safety threshold, providing an adequate safety margin. The average tracking force of 5.72 N indicates generally accurate trajectory following with minor steady-state errors.

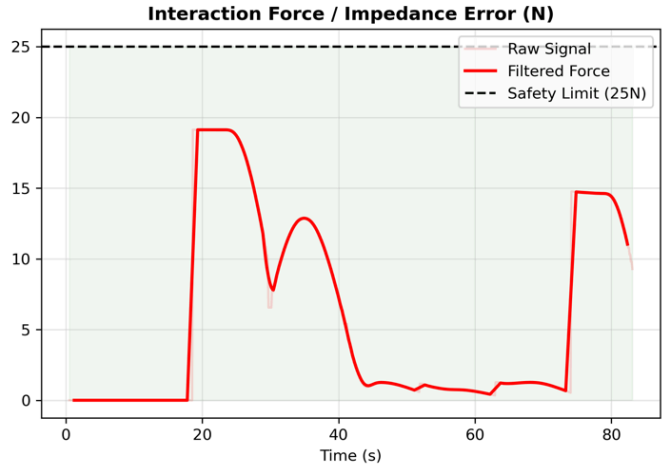

Fig 5: Plot for Interaction Force

3.2.4 Motion Smoothness (Jerk)

The bottom-right panel shows the jerk (third derivative of position) for each joint. Low jerk magnitude indicates smooth motion without abrupt accelerations. The observed jerk magnitudes of approximately ±0.015 rad/s³ are very low, confirming that the trajectory generation and execution produce smooth, comfortable motion suitable for human interaction.

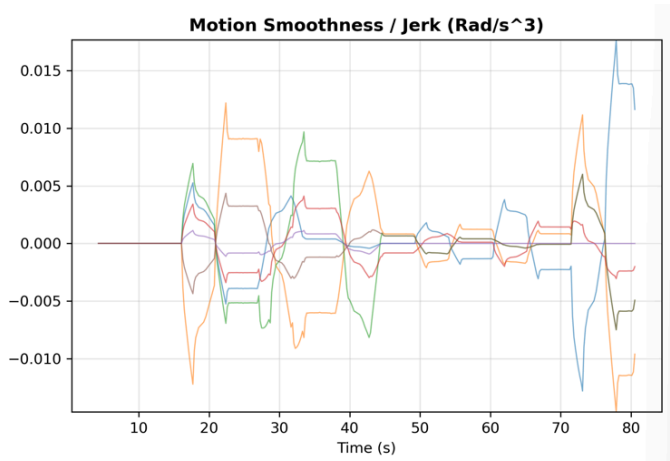

Fig 6: Plot for Motion Smoothness

3.3 Safety System Validation

To validate the safety monitoring system's ability to detect and respond to abnormal conditions, we conducted trials with deliberately challenging initial conditions. Figure 3 shows results from a trial where the robot started from a configuration with large initial error from the first waypoint.

In this trial, the initial motion commanded a large reconfiguration that the robot could not complete within the

expected time, resulting in significant position tracking error. The impedance monitor correctly detected this as a potential collision condition, with estimated force rising to 45.91 N. The safety system triggered an emergency stop, and the robot held its position rather than continuing to exert force.

This result demonstrates two important capabilities. First, the impedance monitoring framework successfully detects conditions that could correspond to collisions or obstructions. Second, the emergency stop mechanism functions correctly, halting motion and preventing further force accumulation.

3.4 Hybrid Planning Validation

We compared trajectory outcomes between pure joint-space planning, pure Cartesian planning, and our hybrid approach. Pure Cartesian planning from the home configuration frequently failed due to singularity conditions when the robot was near its folded initial state the Jacobian becomes ill-conditioned, and the solver cannot find valid paths. Pure joint-space planning succeeded in reaching all waypoints but produced curved end-effector paths during the plate-to-mouth segment, with measured orientation deviations of up to 15 degrees.

The hybrid approach succeeded in all trials, with the joint-space transport phase avoiding singularities and the Cartesian feeding phase maintaining orientation deviations below 2 degrees. This confirms that the hybrid strategy achieves its design goal of combining the strengths of both planning approaches.

3.5 Summary of Quantitative Results

Metric	Value	Threshold/ Target	Status
Total Execution Time	82.51 s	< 120 s	Pass
Peak Joint Velocity	0.148 rad/s	< 0.5 rad/s	Pass
Peak Interaction Force	19.12 N	< 25 N	Pass
Average Tracking Force	5.72 N	< 10 N	Pass
Safety Violations (nominal)	0	0	Pass
Orientation Deviation (feeding phase)	< 2°	< 5°	Pass
Emergency Stop Response	Functional	Required	Pass

Table 2 : Key Performance Metrics across Experimental Trials

IV. CONCLUDING REMARKS

4.1 Summary of Contributions

This work presented a complete simulation framework for assistive feeding robotics, addressing two key challenges: the motion planning trade-off between singularity avoidance and orientation preservation, and the need for safe human-robot interaction without expensive sensing hardware.

The hybrid motion planning strategy resolves the singularity-orientation trade-off by intelligently selecting interpolation methods based on task phase. Joint-space planning provides robust, singularity-free motion during transport phases, while Cartesian planning ensures straight-line, level-spoon motion during feeding phases. The phase-aware IK seeding further improves trajectory consistency by biasing solutions toward kinematically appropriate configurations.

The impedance-based safety monitoring framework demonstrates that meaningful collision detection is achievable without force/torque sensors. By interpreting position tracking errors as virtual spring compressions, the system estimates contact forces in real-time. Adaptive stiffness gains increase's sensitivity near the user's face, and the 25 N emergency stop threshold provides protection consistent with collaborative robot safety standards.

4.2 Lessons Learned

Several technical lessons emerged from this project. First, simulation environments introduce artifacts (such as physics engine noise) that must be addressed through signal processing. The rolling mean filter was essential for preventing false safety triggers. Second, inverse kinematics seeding significantly impacts trajectory quality; phase-aware seeding should be considered standard practice for manipulation tasks. Third, hybrid approaches that select algorithms based on task context outperform single-algorithm approaches that attempt to handle all situations uniformly.

4.3 Limitations

The current implementation has several limitations that should be addressed in future work. The human model is static; real

users move their heads, and the system cannot currently adapt to such motion. The sensorless force estimation cannot distinguish between contact and other sources of tracking error, potentially producing false positives. The simulation has not been validated against physical hardware, and sim-to-real transfer may reveal additional challenges.

4.4 Future Work

Future development should pursue three directions. First, integrating camera-based head tracking would enable dynamic waypoint adjustment as the user moves. Second, deployment on physical UR5e hardware with comparison to force/torque sensor measurements would validate the sensorless estimation approach. Third, implementing true impedance control where estimated forces modulate commanded trajectories in real-time would provide more nuanced interaction than the binary safe/unsafe determination of the current monitoring approach.

REFERENCES

[1] D. Park, Z. Erickson, H. M. Clever, G. Turk, and C. C. Kemp, "Active robot-assisted feeding with a general-purpose mobile manipulator in real homes," *Robotics and Autonomous Systems*, vol. 124, p. 103344, 2020.

[2] N. Hogan, "Impedance control: An approach to manipulation," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 1–24, Mar. 1985.

[3] ISO/TS 15066:2016, "Robots and robotic devices—Collaborative robots," International Organization for Standardization, Geneva, Switzerland, 2016.

[4] T. Bhattacharjee, G. Lee, H. Song, and S. S. Srinivasa, "Towards robotic feeding: Role of haptics in fork-based food manipulation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1485–1492, Apr. 2019.

[5] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, Dec. 2017.

[6]https://drive.google.com/file/d/1xKngBv1I8s_0iQytsXg4q4WPhPXhviom/view?usp=sharing