Sim-to-Real Kitchen Automation: A Robotic System for Cooking Rice

Yicheng Wang, Haotong Han, Yazhou Zhang, Hang Yin, Ruopei Chen
Department of Mechanical Engineering, Stanford University
{wycheng, hthan, zhangyaz, hangyin, rpchen}@stanford.edu

***Abstract*—We present an semi-autonomous robotic system capable of performing the complete workflow for rice cooking, from ingredient collection to cooker activation. The system is built on a mobile manipulator platform equipped with a 7-DOF Franka Emika Panda arm and a RealSense RGB-D D455 camera. Task execution is orchestrated through a hierarchical finite-state machine, which integrates perception, motion planning, and control. Key challenges addressed include accurate vision-guided manipulation, force-compliant interaction with buttons, and safe, repeatable pouring using joint-space control. To ensure smooth arm motion, we designed an interpolated Cartesian controller with finely tuned convergence thresholds and gain parameters. The full task pipeline was first developed and validated in simulation using MuJoCo for vision and manipulation prototyping, and OpenSai for full-stack controller emulation. This simulation-to-reality pipeline enabled safe deployment and reduced debugging time. Our system demonstrates robust multimodal coordination and offers a reproducible framework for automating common household tasks in structured environments.**

***Index Terms*—robotics, mobile manipulation, state machine, human-robot interaction, kitchen automation**

I. Project Description and Objectives

Robotic systems have repeatedly demonstrated their utility in automating tedious and time-consuming human activities. Cooking rice, a process that takes roughly one hour, offers a representative use case. We propose that, by enabling remote control and on-board autonomy, a mobile manipulator could prepare rice in advance so that users return home to freshly cooked meals. Accordingly, the objective of this project is to develop a robot capable of semi-autonomously executing the entire rice-preparation workflow within a semi-structured kitchen environment, leveraging its integrated perception and manipulation capabilities. The task is decomposed into five sequential subtasks: 1) pick up the rice container, 2) get rice from the rice dispenser, 3) pour rice into the cooker, 4) pour water into the cooker, and 5) turn on the rice cooker.

The system is classified as semi-autonomous because high-level behavior is governed by a state machine that presupposes a reasonably structured environment. Nevertheless, the robot's vision pipeline affords sufficient tolerance to variations in the absolute positions of task-relevant objects, providing the robustness needed for everyday kitchen deployment.

II. Robot Description

TidyBot++ (Fig. 1) is selected as the robot platform for this project. It features a Franka Emika Panda arm with 7 degrees of freedom (7-DOF), mounted on a mobile base without holonomic constraints—that is, the base can move omnidirectionally despite its non-holonomic drive system. The end-effector is a Franka Hand gripper equipped with high-friction pads, enabling secure grasps on smooth or slippery kitchenware. For visual perception, an Intel RealSense D445 RGB-D camera is mounted near the wrist to provide RGB and depth information. This configuration supports fiducial marker detection, object localization, and perception-guided manipulation in semi-structured kitchen environments.

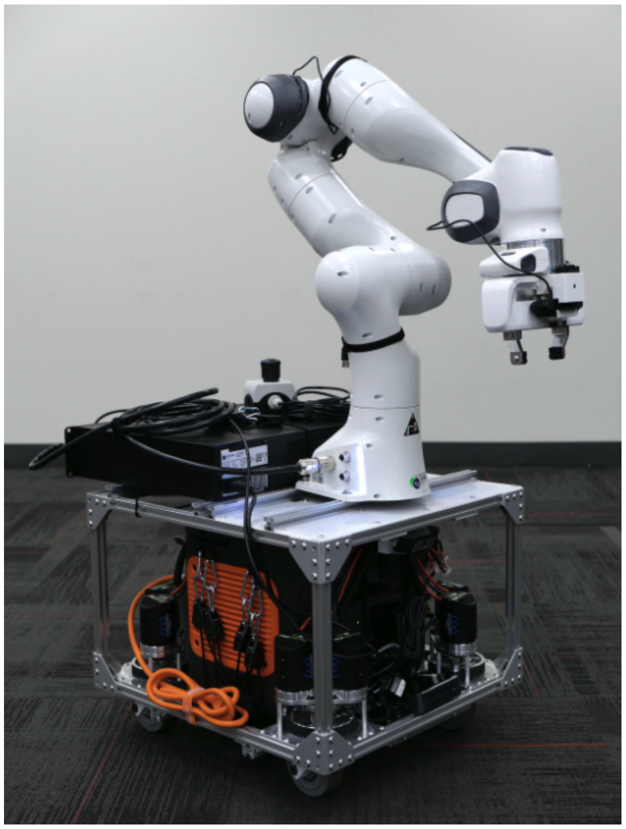


Fig. 1. TidyBot++ system composed of a Franka Emika Panda arm, gripper, RGB-D camera, and mobile base.

III. Simulation Environment

To ensure hardware trials begin from safe and collision-free configurations, we developed a dual-simulator pipeline to support environment modeling, perception tuning, and controller

This project was completed for Stanford CS225A: Experimental Robotics.

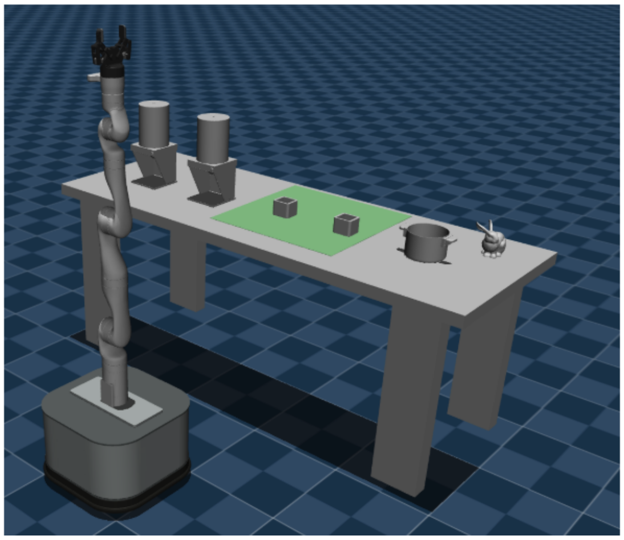


Fig. 2. MuJoCo simulation environment used for rapid prototyping of perception and manipulation tasks.

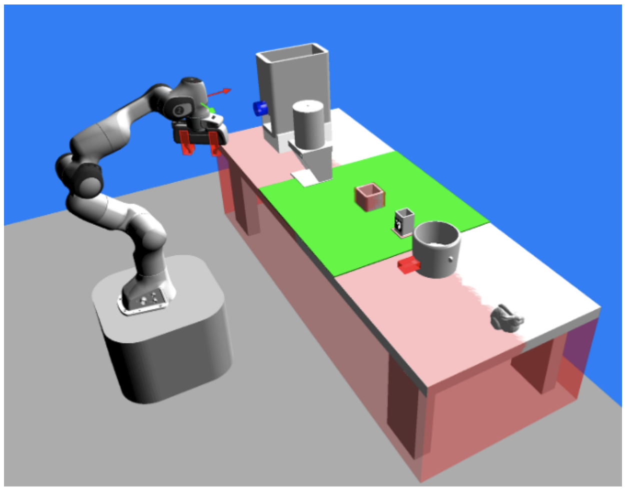


Fig. 3. OpenSai simulation environment replicating full-stack control with joint and impedance controllers.

validation. Two complementary simulators—MuJoCo (Fig. 2) and OpenSai (Fig. 3)—served distinct yet synergistic roles in our system development process.

MuJoCo: Lightweight and Vision-Driven Prototyping: We leveraged MuJoCo for its computational efficiency and ease of integration with custom camera pipelines. Each object relevant to the rice-cooking task—including dispensers, containers, and the cooker—was equipped with ArUco markers. This setup allowed us to repeatedly test and refine our perception algorithms across varying lighting and viewpoint conditions. Cartesian trajectory scripts were evaluated in this environment, enabling us to tune interpolation steps, control gains, and grasp approach vectors.

OpenSai: Control-Accurate and Behavior-Faithful Validation: Once perception and manipulation strategies were stable, they were ported into OpenSai, which replicates the entire

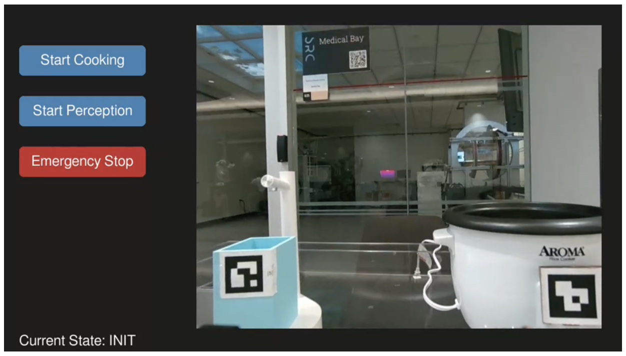


Fig. 4. Graphical user interface (GUI) developed in Pygame for controlling perception, task execution, and emergency stop.

real-robot control stack. OpenSai models joint-level actuation, Cartesian impedance, and control switching behavior through Redis interfaces. This made it possible to rehearse full-task sequences—such as grasp-pour-place routines—with fidelity to hardware timing, latencies, and low-level safety logic. Our finite state machine (FSM), force-based control logic, and trajectory interpolation modules were validated here under realistic robot behavior.

Physical Modeling and Dynamic Fidelity: For both platforms, we built custom CAD models of the table, robot, and all interactive objects. These models were updated iteratively to reflect actual measurements. Special attention was paid to collision geometry and inertial properties, which were hand-tuned to promote stable physics interactions.

One key challenge was grasping simulation. Dynamic modelling inaccuracies led to oscillations or slippage in grasped objects. To mitigate this, we introduced a virtual "linked" mode: once a grasp is detected (e.g., finger contact confirmed), the object's pose is rigidly attached to the end-effector frame, bypassing unstable contact dynamics.

Simulation-Guided Development: The simulation environments allowed us to test hundreds of variations in grasp height, joint configurations, and control parameters without wear or risk to hardware. For example, workspace feasibility for pouring actions was analyzed entirely in simulation, ensuring reachable and collision-free motions before real-world trials. Force thresholds for compliant pressing were also tuned virtually using logged end-effector trajectories and displacements.

Together, MuJoCo and OpenSai enabled a robust simulation-to-reality pipeline. MuJoCo offered fast vision-algorithm development and trajectory experimentation, while OpenSai supported system-level integration and control validation. This layered approach helped us preempt many failure modes and reduce real-world debugging time substantially.

IV. Human Interface

We developed a lightweight graphical user interface (GUI) using Python's `pygame` library to simplify control over the autonomous cooking system. Upon launch, the interface initializes essential backend drivers, including those for the

Franka Emika Panda arm and the mobile base, ensuring the robot is fully operational before any task begins.

The GUI provides three primary buttons:

- **Open Perception:** Activates the RGB-D camera feed in a separate, docked window for real-time visual monitoring.
- **Start Cooking:** Triggers the full task sequence, including navigation, grasping, pouring, and pressing actions as governed by the finite-state machine.
- **Emergency Stop:** Immediately halts all robot actions for safety or manual intervention.

This interface prioritizes clarity, usability, and state-based control, enabling both developers and operators to manage system execution with minimal overhead.

Looking ahead, we plan to extend this system to support remote task initiation via smartphone over WiFi and to expand the underlying task library to include additional household automation functions. With the combined mobility and perception capabilities, the robot could further serve as a mobile monitoring unit, streaming real-time household status to remote users.

Fig. 4 shows the GUI interface with labeled controls.

V. State Machine and Controllers

The autonomous rice cooking task is governed by a hierarchical finite-state machine (FSM), shown in Fig. 5, which coordinates mobile navigation, arm manipulation, perception, and error handling. Each FSM state corresponds to a well-defined subtask (e.g., pick, pour, press), with transitions triggered by success flags, retries, or sensor feedback.

Controller Architecture: Multiple low-level controllers are deployed dynamically depending on the subtask's physical requirements:

- **End-Effector Cartesian Control:** The majority of manipulation tasks (grasping, transporting, placing) are executed using position-based Cartesian control. To ensure smooth and stable motion, we fine-tuned the convergence thresholds for both position and orientation (e.g., 1–2 cm, $< 7°$) and carefully adjusted the controller's proportional and derivative gains (k_p, k_v). These parameters were optimized to minimize steady-state error and overshooting while maintaining responsiveness. Furthermore, we implemented trajectory interpolation to add intermediate waypoints, sending each new command only after the previous one was successfully reached. This strategy helped avoid jerky transitions, particularly when transporting filled containers, and preserved the upright orientation of objects during motion.
- **Force-Based Button Pressing:** Due to inherent uncertainties in surface compliance and joint friction, pose-based control often failed to reliably press buttons. We implemented a task-space force control scheme to address this. A forward-directed force is applied to overcome internal resistance, and end-effector motion is monitored. When forward motion stagnates, contact is assumed; we then apply a transient force pulse to ensure activation, followed by a lower holding force to maintain button contact. This staged force profile enables robust interaction without slipping or oscillation.
- **Joint Space Pouring Control:** Pouring actions are executed using joint space control of the final revolute joint (wrist). We regulate the motion via proportional-derivative (PD) control, tuning k_p and k_v to achieve smooth and repeatable pouring dynamics. A predefined angular increment (e.g., $105°$) is applied during the pour, and reversed after a short delay to reset the posture. This approach reduces orientation noise and local instability compared to Cartesian-based pouring.
- **Mobile Base Control:** The mobile base is position-controlled via velocity commands to target waypoints. No active localization is used; instead, open-loop odometry is sufficient given structured indoor scenes and short travel distances.
- **Gripper Control:** The Franka Hand gripper operates in discrete open/close modes. Grasping occurs only after convergence to a verified pose, and grasped objects are virtually coupled to the end-effector in software to mitigate simulation artifacts.

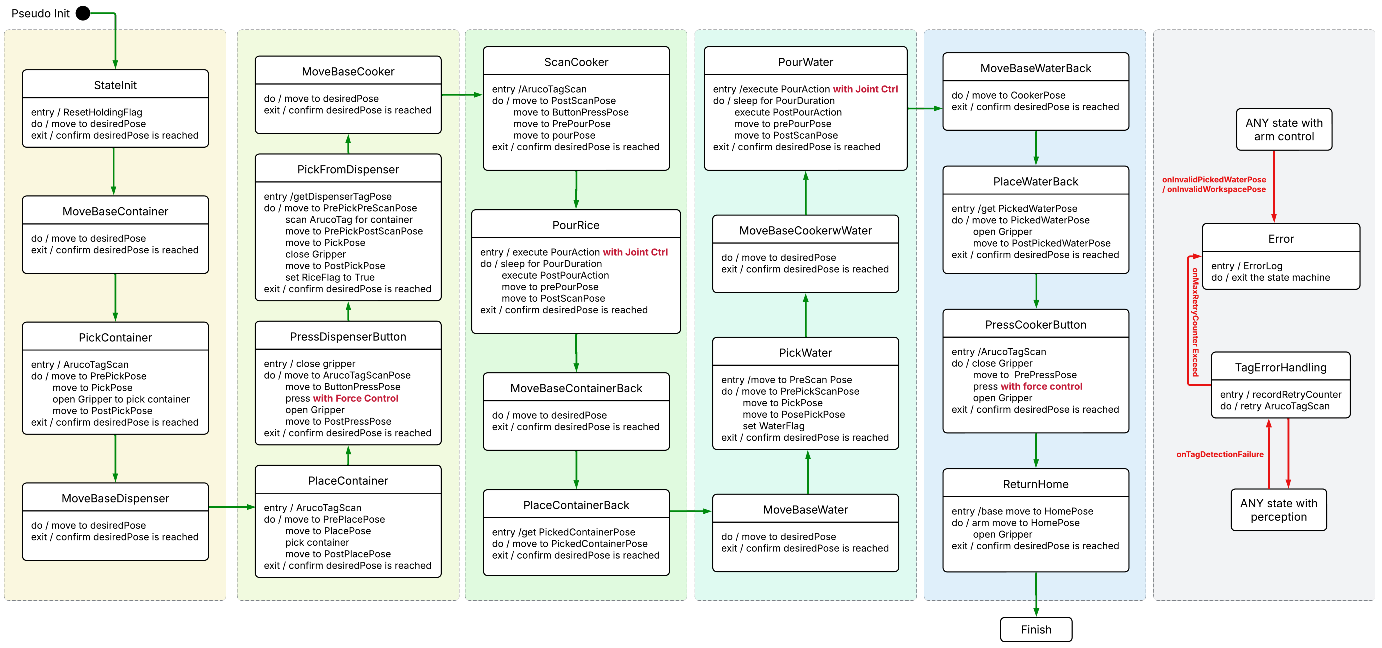
Null-Space Strategy for 7-DOF Redundancy: While Cartesian tasks require only 6 degrees of freedom (DoF), the Franka arm has 7 DoF, introducing redundancy. Without constraint, this null space can lead to joint drift toward limits, which risks controller faults or driver shutdown. To mitigate this, we manually initialize joint configurations near the center of their allowable ranges. This symmetric initialization reduces the chance of encountering joint limits mid-task, ensuring kinematic feasibility throughout long-horizon tasks.

Perception-Driven Transitions and Error Handling: ArUco tag-based localization is used for all perception-guided tasks. Tag pose (position + orientation) is transformed to the base frame using hand-eye calibration. A fusion step compares raw ArUco center estimates with depth-fused 3D centroids; fusion error above threshold (e.g., 4 cm) triggers re-scan. The FSM includes dedicated error states to handle perception failures, retry limits, and recovery strategies.

VI. Challenges and Future Work

Challenges: Despite the success of the autonomous rice cooking system, several challenges were encountered during development:

- **Grasping Instability in Simulation:** In OpenSai, the simulated contact dynamics between the gripper and objects were not always reliable. This resulted in oscillations or slippage during grasp attempts. To address this, as long as the gripper finger closes, we keep updating the positions of grasped containers in `simviz.cpp` to be the same positions as the gripper central point. Hence in visualization, we can simulate the grasping, lifting, placing and pouring of containers.
- **Perception Sensitivity:** The ArUco-based perception pipeline was occasionally affected by occlusions, depth



Pseudo Init
StateInit
entry / ResetHoldingFlag
do / move to desiredPose
exit / confirm desiredPose is reached
MoveBaseContainer
do / move to desiredPose
exit / confirm desiredPose is reached
PickContainer
entry / ArucoTagScan
do / move to PrePickPose
move to PickPose
open Gripper to pick container
move to PostPickPose
exit / confirm desiredPose is reached
MoveBaseDispenser
do / move to desiredPose
exit / confirm desiredPose is reached
MoveBaseCooker
do / move to desiredPose
exit / confirm desiredPose is reached
PickFromDispenser
entry /getDispenserTagPose
do / move to PrePickPreScanPose
scan ArucoTag for container
move to PrePickPostScanPose
move to PickPose
close Gripper
move to PostPickPose
set RiceFlag to True
exit / confirm desiredPose is reached
PressDispenserButton
entry / close gripper
do / move to ArucoTagScanPose
move to ButtonPressPose
press with Force Control
open Gripper
move to PostPressPose
exit / confirm desiredPose is reached
PlaceContainer
entry / ArucoTagScan
do / move to PrePlacePose
move to PlacePose
pick container
move to PostPlacePose
exit / confirm desiredPose is reached
ScanCooker
entry /ArucoTagScan
do / move to PostScanPose
move to ButtonPressPose
move to PrePourPose
move to pourPose
exit / confirm desiredPose is reached
PourRice
entry / execute PourAction with Joint Ctrl
do / sleep for PourDuration
execute PostPourAction
move to prePourPose
move to PostScanPose
exit / confirm desiredPose is reached
MoveBaseContainerBack
do / move to desiredPose
exit / confirm desiredPose is reached
PlaceContainerBack
entry /get PickedContainerPose
do / move to PickedContainerPose
exit / confirm desiredPose is reached
PourWater
entry /execute PourAction with Joint Ctrl
do / sleep for PourDuration
execute PostPourAction
move to prePourPose
move to PostScanPose
exit / confirm desiredPose is reached
MoveBaseCookerwWater
do / move to desiredPose
exit / confirm desiredPose is reached
PickWater
entry /move to PreScan Pose
do / move to PrePickScanPose
move to PickPose
move to PosePickPose
set WaterFlag
exit / confirm desiredPose is reached
MoveBaseWater
do / move to desiredPose
exit / confirm desiredPose is reached
MoveBaseWaterBack
do / move to CookerPose
exit / confirm desiredPose is reached
PlaceWaterBack
entry /get PickedWaterPose
do / move to PickedWaterPose
open Gripper
move to PostPickedWaterPose
exit / confirm desiredPose is reached
PressCookerButton
entry /ArucoTagScan
do / close Gripper
move to PrePressPose
press with force control
open Gripper
exit / confirm desiredPose is reached
ReturnHome
entry /base move to HomePose
do / arm move to HomePose
open Gripper
exit / confirm desiredPose is reached
Finish
ANY state with arm control
onInvalidPickedWaterPose / onInvalidWorkspacePose
Error
entry / ErrorLog
do / exit the state machine
onMaxRetryCounter Exceed
TagErrorHandling
entry / recordRetryCounter
do / retry ArucoTagScan
onTagDetectionFailure
ANY state with perception

Fig. 5. Hierarchical finite-state machine for autonomous rice cooking. Each color block denotes a task phase; red arrows indicate error transitions; red text highlights control mode switches.

noise, or poor lighting, leading to localization errors. While retry logic and scan positions helped mitigate this, future implementations could benefit from more robust perception algorithms, such as depth-fused pose estimation or learned object keypoint detection.

- **Force Control Complexity:** Button-pressing tasks required delicate force-based control to ensure contact detection, force modulation, and safety. Tuning parameters such as contact thresholds and pressing duration proved nontrivial and highly context-dependent.
- **Null Space Drift and Joint Limits:** The 7-DOF Franka arm's redundancy introduced challenges in maintaining safe joint configurations. Without null-space optimization, joints occasionally approached their limits during motion, risking controller shutdown. This was mitigated by initializing joints near the center of their ranges, though more sophisticated null-space policies are needed for long-horizon tasks.

Future Work: Future extensions will focus on increasing robustness, generality, and system autonomy:

- **Incorporate real-world dynamics into the hardware demo:** Because of time constraints, the current demonstration omits real rice and water. In future iterations, introducing actual rice and water will let us capture and account for the pouring dynamics during the dispensing phase.
- **Wi-Fi Remote Control with Semantic Task Execution:** By using a smartphone app on the same Wi-Fi network as the robot, users can issue voice or text commands such as "cook rice." The app converts each command into a semantic task plan such as rice preparation or any other predefined recipe, and sends it over the local network to the robot's controller. In practice, a student working in the SRC can simply say "cook rice" on their phone and by the time they return home a hot meal will be ready.
- **Improved Perception and Object Tracking:** Incorporating object tracking, depth fusion, or neural pose estimation could help disambiguate objects in cluttered scenes and handle partial occlusions. By doing so, we could deploy the task in a more complex environment.
- **Null-Space and Motion Optimization:** Introducing null-space projection techniques or redundancy resolution algorithms (e.g., manipulability maximization or obstacle-aware joint-space optimization) can improve motion safety and reduce the risk of singularities or joint-limit violations.
- **Expanded Task Library:** The current FSM can be extended to include broader kitchen tasks such as stirring, ingredient fetching, or multi-container operations—laying the groundwork for a general-purpose household assistant.

VII. Demo Video Links

https://drive.google.com/file/d/1rpZnk57MXJJ5K2RYl7KXihkC4oyoOSC7/view?usp=sharing