FridgeGenie

A1: Steven Shi, Neo Jun Wei, Will Chen
18-500 Capstone Design, Spring 2025
Electrical and Computer Engineering Department
Carnegie Mellon University

Product Pitch

Finding and using groceries is a major pain point for households, especially busy college students. Hectic schedules often lead to expired food and forgotten ingredient, and manually keep tracking of your groceries is tedious and inconvenient.

Our project, FridgeGenie, solves this by introducing an intelligent, **vision-driven inventory management** system. We combine **physical mounted cameras** with **computer vision object detection** and **mobile app integration**, helping users automatically detect and classify groceries.

Our solution managed to achieve a **item detection accuracy of ~80%** with an **average inference time of under 1s**. The captured images are stitched, uploaded and sent through the CV pipeline **without manual labeling** from users. Grocery lists and recipe recommendations can be generated in **less than 5 seconds** based on real-time inventory.

System Architecture

Our system uses a Raspberry Pi as the central coordinator, interfacing with both the camera hardware and our cloud pipeline. A fridge-mounted system, which consists of an IMX219-77 camera, LED ring light, and a motorized camera slider driven by a 12V stepper motor and A4988 driver, captures sequential images of fridge content. Captured images are transmitted wirelessly to an AWS cloud server, where our trained CV model processes the data to detect and classify grocery items. The results are stored in a cloud-based inventory database alongside a recipe database. The mobile app is able to trigger fridge scans, as well as utilize the databases for recipe recommendations and grocery list generation.

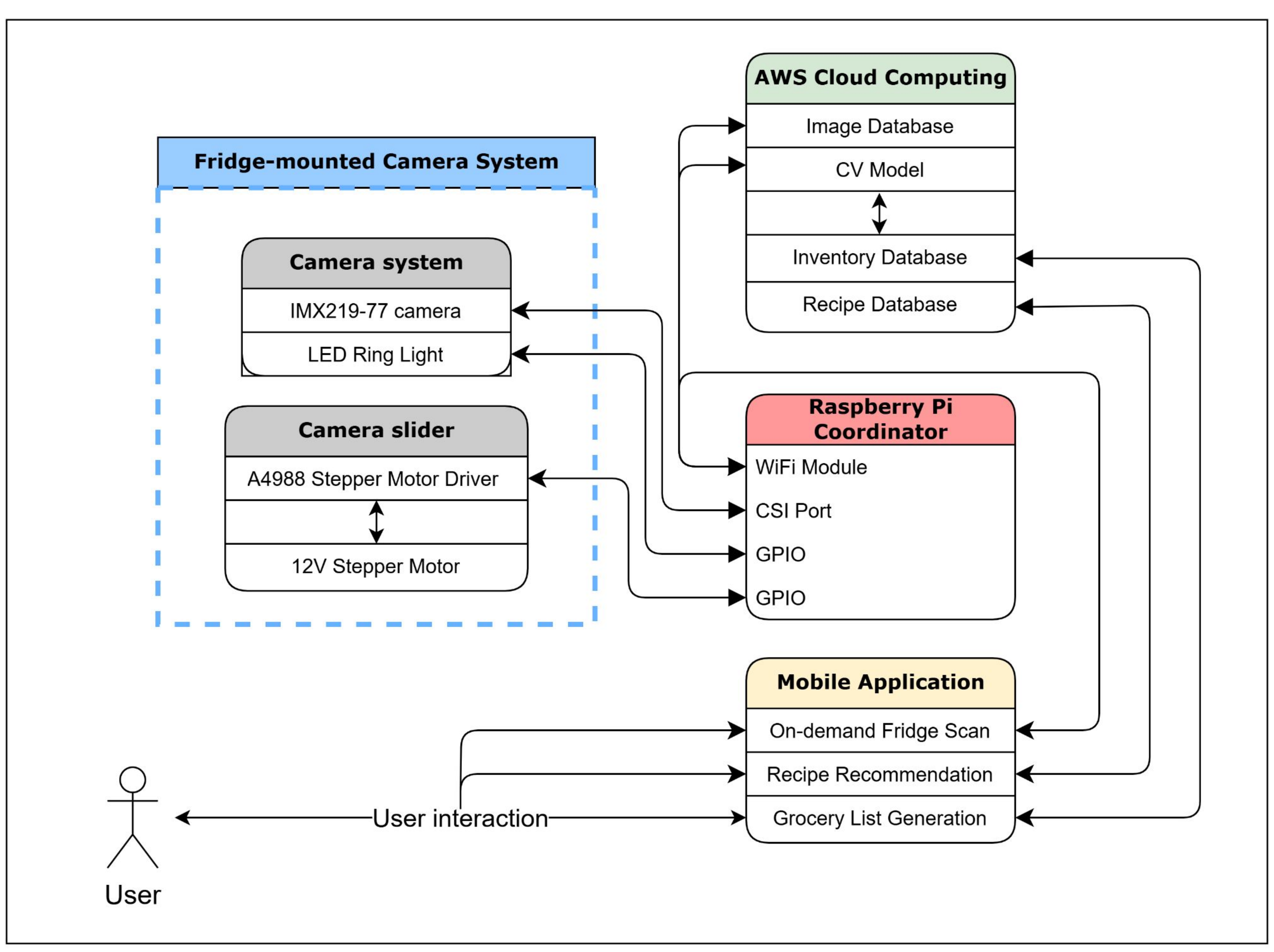

Figure 1: Block Diagram of System

Conclusions & Additional Information

Our goal for FridgeGenie was to create an automated, user-friendly and reliable grocery inventory management system. Our final implementation was closely aligned with our original vision, as we managed to create a motorized camera slider, consistent lighting control, cloud-based CV inference and a mobile application for user interaction. Our system meets the core user requirements by providing automated fridge scans, accurate grocery detection and recipe recommendations.

During the process, we encountered challenges that taught us valuable lessons. Delays in hardware shipping and incompatibility between our initial hardware components emphasized the importance of contingency planning and modular development. Fine-tuning our CV model also made us realize the importance of high-quality datasets as opposed to simply scaling model size.

https://course.ece.cmu.edu/~ece500/projects/s25-teama1/

System Description

Fridge-Mounted Camera System

Our camera system consists of a an IMX219-7 camera mounted on a motorized rail within the bridge. The camera captures sequential high-resolution images during each scan cycle, which are then stitched together to ensure full coverage of fridge contents. The camera is connected to the Raspberry Pi via the CSI port for fast image transfer.

A 12V Stepper Motor is used to move the camera horizontally across the fridge, the motor itself being driven by an A4988 stepper motor driver controlled by the Raspberry Pi using GPIO PWM signals. The camera is calibrated to stop at predefined interval for image capture, ensuring minimal image overlap when stitching. A white LED ring light is used to provide consistent lighting for image capture.

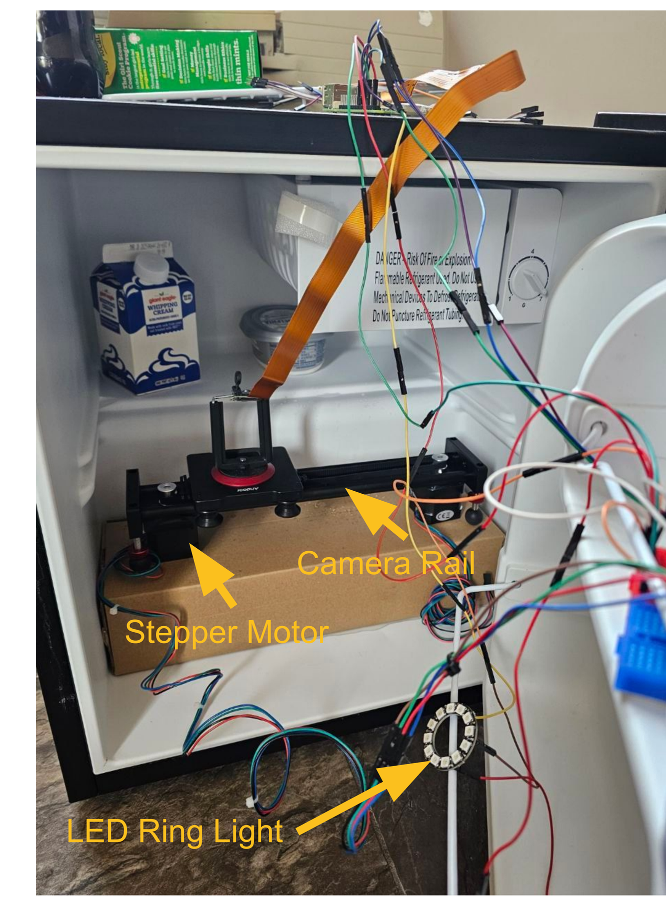

Figure 2: Mounted Camera System

CV Model

Our CV model is based on the YOLOv5x model, and was fine-tuned for grocery item detection. The model was trained on several grocery image datasets including Open Images for food categories. We utilized transfer learning starting from pretrained COCO weights and fine-tuned the model on our custom dataset.

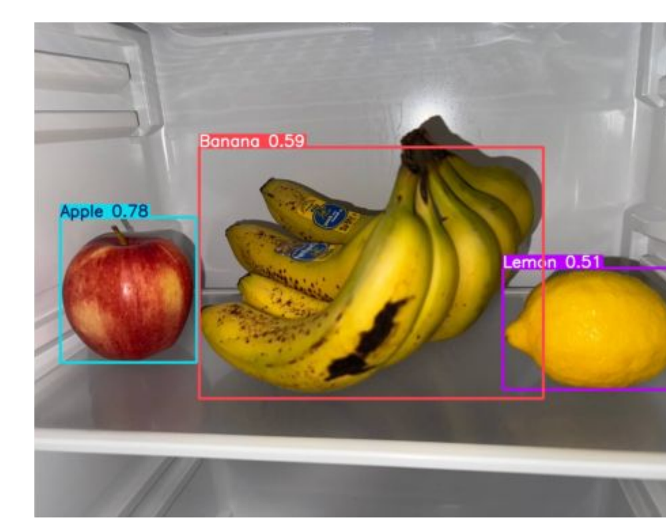

Figure 3: Output of CV Model

Mobile App

Our mobile app allows for on-demand fridge scans and accurately displays identified fridge contents. It communicates securely with our backend cloud database to retrieve real-time data and inventory status, while also enabling manual item corrections in the case of misclassification.

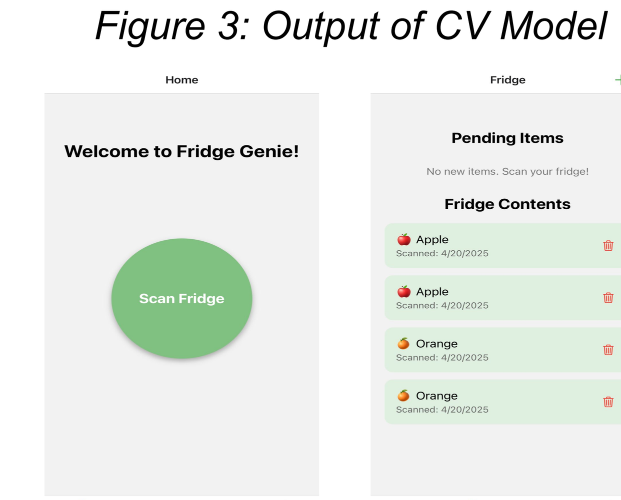

Figure 4: Mobile App UI

System Evaluation

Use-Case Requirements

- Overall workflow should complete within a reasonable time to ensure usability
- High model accuracy to reduce users having to manually correct items
- Functional recipe and grocery recommendations

Design Requirements

Metric	Target	Actual
Camera Scan Latency	5s	8-9s
Stitched Image Latency	3s	5-6s
Object Scan Latency	2s	~500ms
Object Detection Accuracy	90%	~80%
Recipe Recommendation Latency	5s	<5s
Grocery List Generation Latency	5s	<5s

A design trade-off we had was the accuracy and latency of the YOLO model we used, and also whether to run inference locally on the Raspberry Pi or the cloud. We initially experimented with running inference on the Raspberry Pi, however inference time increased dramatically, which made larger models unfeasible. Ultimately, we decided to make use of cloud-based inference, allowing us to use the largest YOLOv5x model to optimize our accuracy while remaining within our latency requirements.

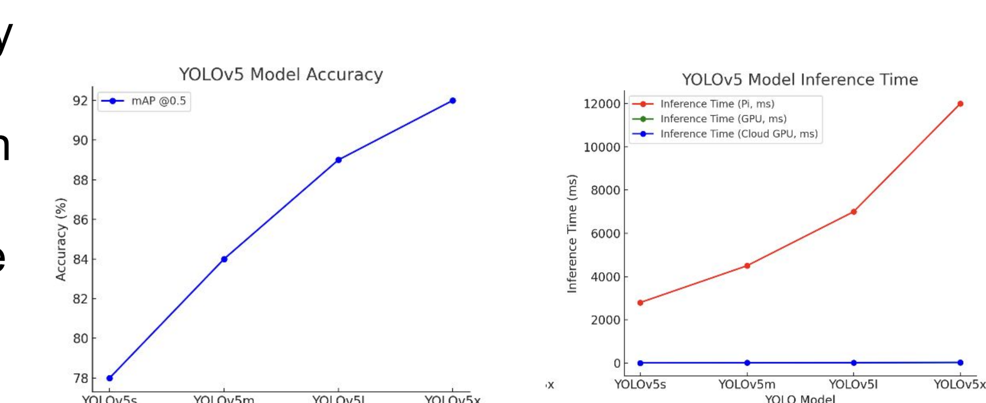

Figure 5: Inference Accuracy and Latency for YOLOv5 Models