NeuroGrip

Autonomous Reshelving using Vision Language Models

Yu-Hsin (Thomas) Chan, Boxiang (William) Fu, Joshua Pen, and Jet Situ

Robot Autonomy Team 3A
Carnegie Mellon University

Motivation

Why do existing robots find it difficult to work in a versatile and dynamic environment?

- **Difficulty in generalized visual recognition and interpretation**
- **Complexity of working with dynamic distances in a workspace**
- **Novelty of visual servoing techniques on a reliable scale**

A generalized workspace robot would provide a significant leap in potential for both a wide range of commercial and industrial applications.

Problem Statement

Given a robot and several objects, demonstrate generalized pick and place tasks.

1. The robot must be able to autonomously recognize and locate specified objects within a scene.
2. The robot must then autonomously move to the object.
3. Pick up the object, and then place them in a designated location. For generalization, typical industrial and commercial approaches should be avoided - no AR tags, no guidewires, no pre-positioning of the pick positions, etc.

Approach

Components:

- **Vision-Language Model (VLM):** Utilizes GPT API to convert text prompts into actionable commands for the robotic system.

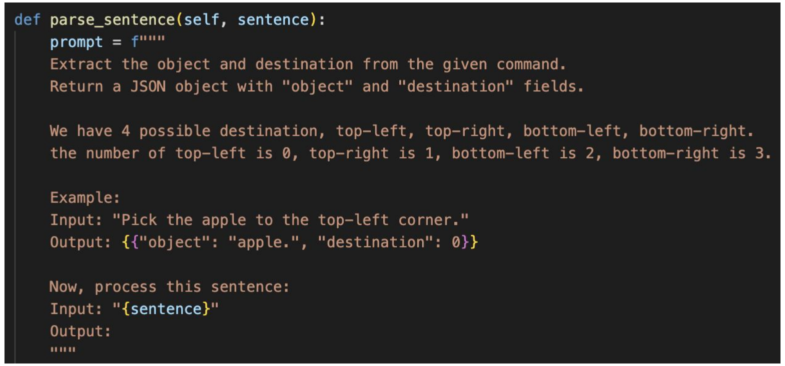
- **Intel RealSense Camera:** Captures depth and RGB data to identify and locate target objects within the environment.

- **Robotic Arm:** Executes pick-and-place tasks based on commands derived from the VLM and positional data from the camera.

Text Prompt Interpretation

- User provides a text prompt specifying the desired object to be picked up and the drop-off location (e.g., "top left," "top right," "bottom left," "bottom right" of the shelf).

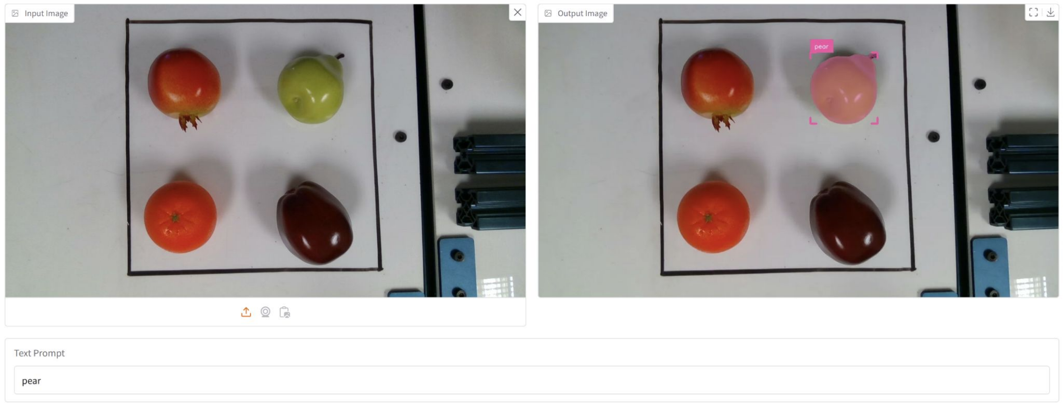
- The GPT-based VLM processes the prompt to determine the target object and the specified drop-off location, generating corresponding action commands.



```
def parse_sentence(self, sentence):
    prompt = f"""
    Extract the object and destination from the given command.
    Return a JSON object with "object" and "destination" fields.

    We have 4 possible destination, top-left, top-right, bottom-left, bottom-right.
    the number of top-left is 0, top-right is 1, bottom-left is 2, bottom-right is 3.

    Example:
    Input: "Pick the apple to the top-left corner."
    Output: {{"object": "apple.", "destination": 0}}

    Now, process this sentence:
    Input: "{sentence}"
    Output:
    """
```

Object Identification and Localization

- The RealSense camera captures the scene, providing depth and RGB images.
- Image processing techniques are applied to detect and segment objects within the camera's field of view.
- The system calculates the center of each detected object by averaging the mask of each item, estimating their positions in the camera frame.

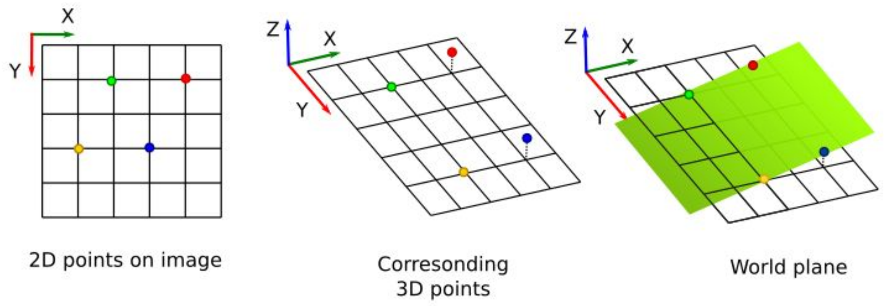


Text Prompt

pear

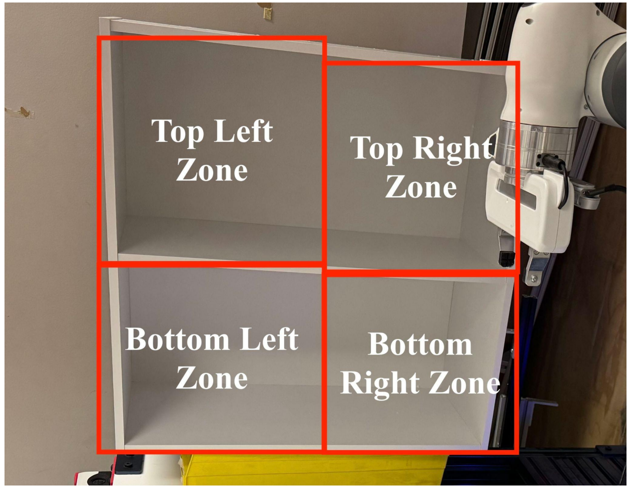
Calibration and Position Estimation

- The robotic arm is calibrated at different positions within the pickup zone to establish a reference frame.
- The positions of objects in the camera frame are mapped to the robot's coordinate system using the calibration data.
- This mapping allows the system to estimate the approximate position of each object relative to the robotic arm.



Drop-Off Location Specification

- The shelf is divided into predetermined zones: top left, top right, bottom left, and bottom right.
- Each zone corresponds to specific coordinates in the robot's workspace, established during the system setup.
- The VLM interprets the specified drop-off location from the text prompt and translates it into the corresponding coordinates.



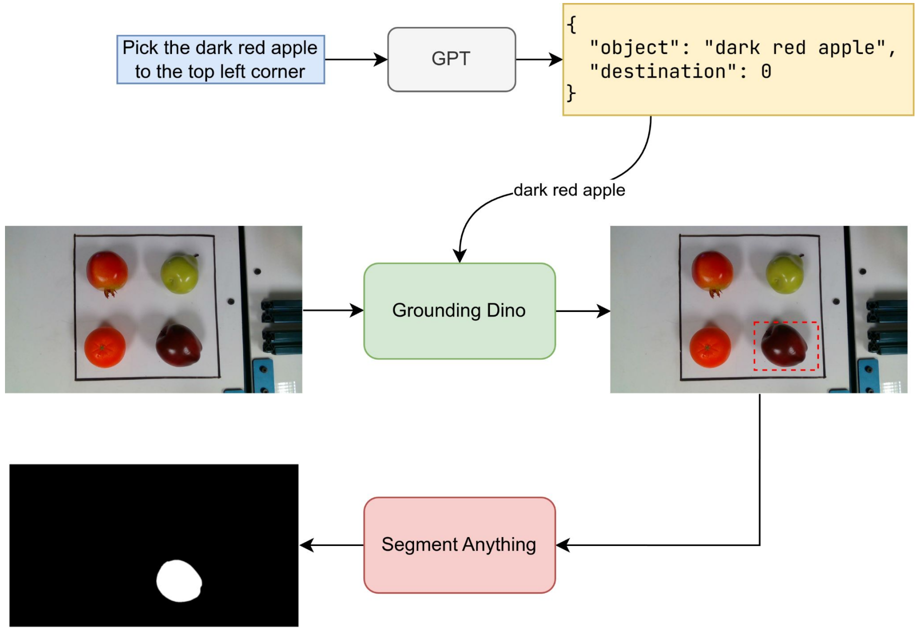
Position Control and Object Manipulation

- The estimated position of the target object is set as the desired position in the robotic arm's position controller for pickup.
- The controller adjusts the arm's movements to align with the target position, enabling accurate pickup of the object.
- The robotic arm then moves to the predetermined drop-off location coordinates and places the object accordingly.

Minimal Viable Product Implementation

Completed:

- Text Prompt Interpretation
- Object Identification and Localization
- Calibration and Position Estimation
- Drop-Off Location Specification
- Position Control and Object Manipulation
- Stretch Goals (see later slides)



```
{
  "object": "dark red apple",
  "destination": 0
}
```

MVP Implementation

https://youtu.be/WbBWrO_oRvI

Blockers: OpenVLA Implementation

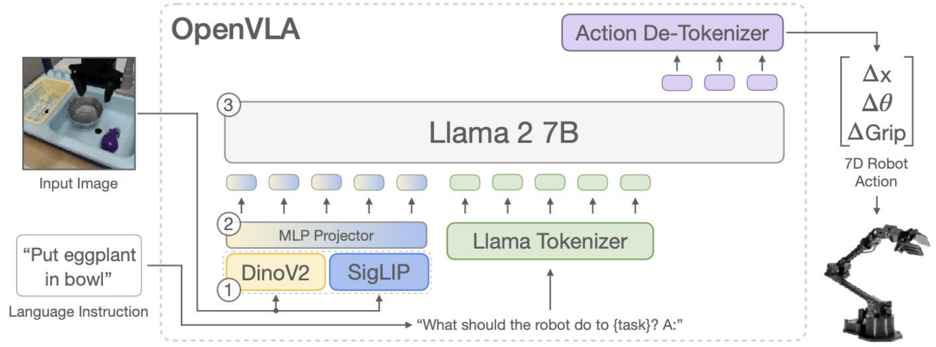
Approach: Tried implementing pre-trained end-to-end OpenVLA framework to directly obtain robot arm Cartesian control commands.

Challenges:

- Model uses Llama 2 7B under the hood and requires 16 GB of VRAM. The control PC has insufficient VRAM.
- Model is trained on WidowX arm, need to further tune to work on Franka arm.

Resolution: Gave up on approach and tried using TinyVLA.

Implementation: Could not get it to work.



Blockers: TinyVLA Implementation

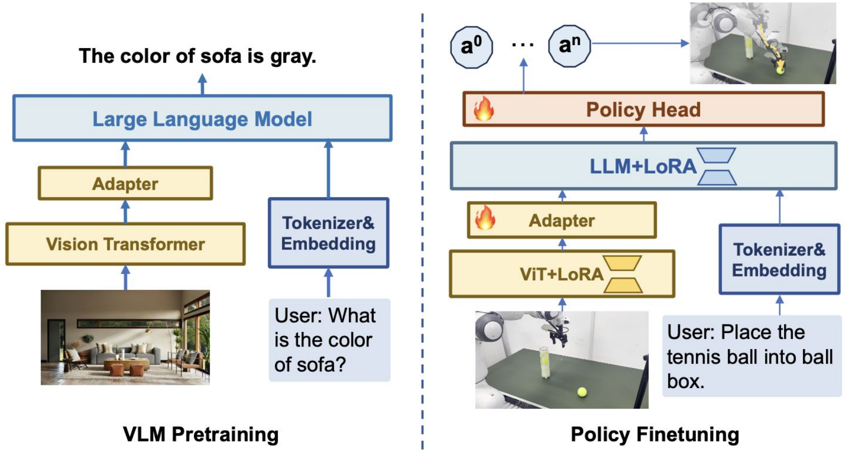
Approach: Tried implementing pre-trained end-to-end TinyVLA framework to directly obtain robot arm Cartesian control commands.

Challenges:

- Very poor deployment documentation.
- Used ZED 2i stereo camera, will need to further tune to make it RealSense compatible.
- General performance is poor (31.6% success rate).

Resolution: Gave up on approach and tried using traditional method.

Implementation: Did not implement.



Blockers: Our Implementation

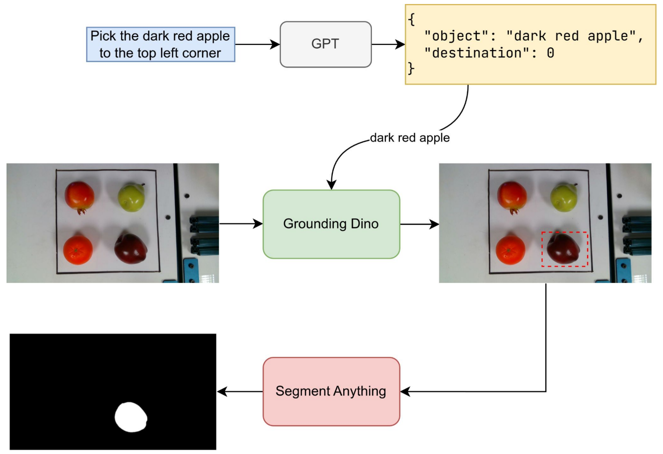
Approach: GPT parses prompt to give "object". Uses Grounding DINO and Segment Anything 2 to obtain Cartesian location of item.

Challenges:

- CUDA version on lab control PC was outdated and couldn't run the required models.

Implementation: Implemented successfully.

Resolution: Established two-way connection between lab control PC and Thomas' home PC. RealSense camera sends image to home PC to process, and sends back Cartesian location.



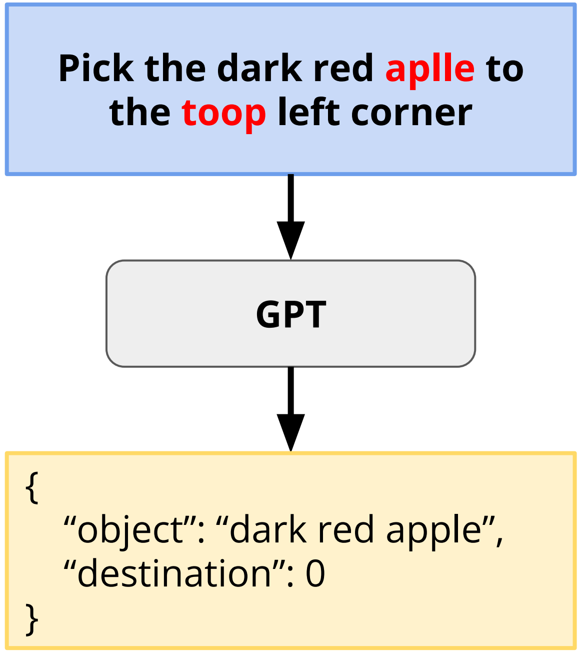
```
{
  "object": "dark red apple",
  "destination": 0
}
```

Plans: Stretch Goal 1 - Spelling Error

Desirables: Have the model understand slight spelling mistakes and still pick up the correct item and move to the correct location.

Implementation: Use a GPT model that is capable of this.

Status: Implemented using OpenAI GPT-4o mini model.



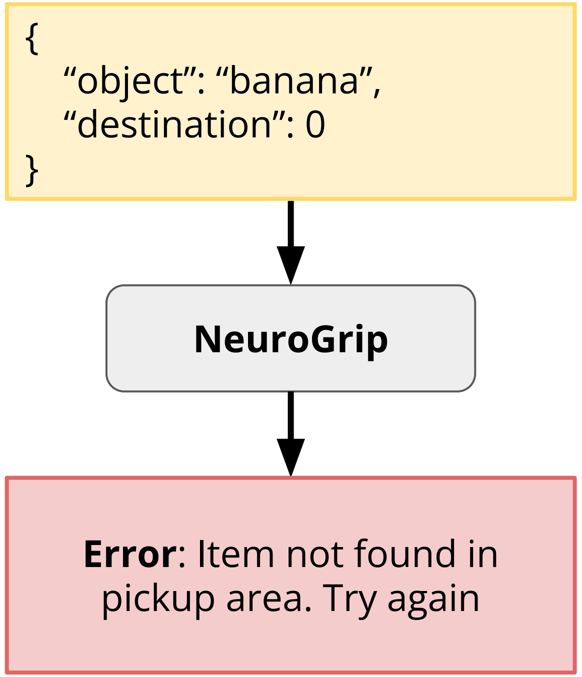
```
{
   “object”: “dark red apple”,
   “destination”: 0
}
```

Plans: Stretch Goal 2 - No Valid Item

Desirables: Have the model reject picking up items if the item does not exist in the pickup zone.

Implementation: Implement a similarity threshold so that only items above the threshold are picked up.

Status: Implemented and tuned Segment Anything 2 item similarity threshold (final threshold value of 0.3).



```
{
    "object": "banana",
    "destination": 0
}
```

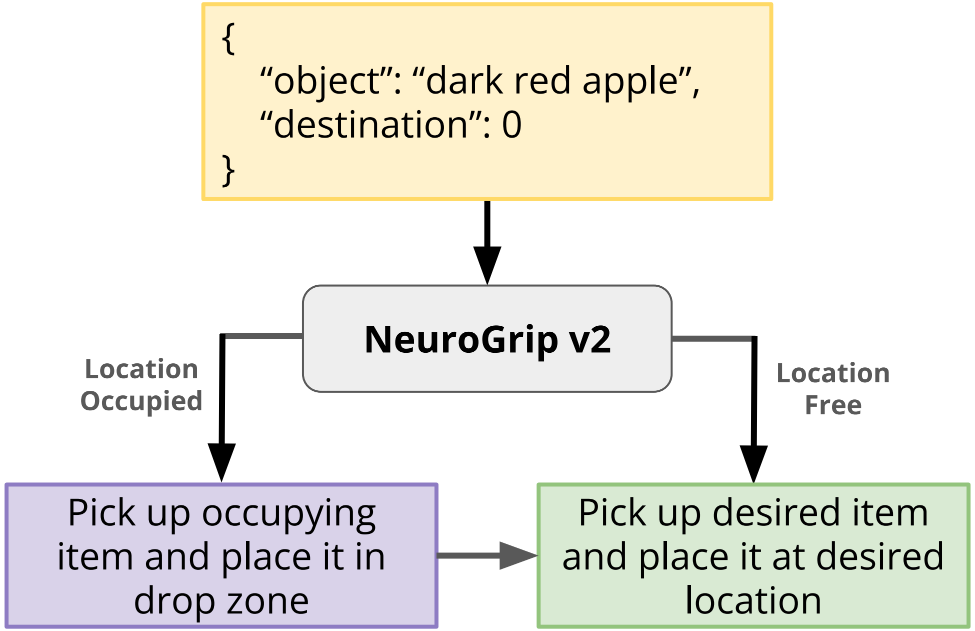
Error: Item not found in pickup area. Try again

Plans: Stretch Goal 3 - Occupied Shelf

Desirables: Have the model check if the desired placement location is occupied. If it is, remove the occupying item first before picking up the desired pickup item.

Implementation: Implement an initial lookup of the shelf to see if location is occupied. If it is, first pick up the occupying item and put it in the drop zone. Return to the pickup zone to pick and place the desired item.

Status: Implemented.



{
"object": "dark red apple",
"destination": 0
}
NeuroGrip v2
Location Occupied
Location Free
Pick up occupying item and place it in drop zone
Pick up desired item and place it at desired location

Final Implementation

```
[illegible]
```

https://youtu.be/2TF8aYAhmj0

Thank You!

https://github.com/NeuroGrip